ACODV: Ant Colony Optimisation Distance Vector Routing in Ad hoc Networks

by

Johan du Plessis

Submitted in partial fulfillment of the requirements for the degree
Magister Scientiae (Computer Science)
in the Faculty of Engineering, Built-Environment
and Information Technology
University of Pretoria
South Africa
November 2005

ACODV: Ant Colony Optimisation Distance Vector Routing in Ad Hoc Networks

by

Johan du Plessis

Abstract

A mobile ad hoc network is a collection of wireless mobile devices which dynamically form a temporary network, without using any existing network infrastructure or centralised administration. Each node in the network effectively becomes a router, and forwards packets towards the packet's destination node. Ad hoc networks are characterized by frequently changing network topology, multi-hop wireless connections and the need for dynamic, efficient routing protocols.

This work considers the routing problem in a network of uniquely addressable sensors. These networks are encountered in many industrial applications, where the aim is to relay information from a collection of data gathering devices deployed over an area to central points. The routing problem in such networks are characterised by:

- The overarching requirement for low power consumption, as battery powered sensors may be required to operate for years without battery replacement;
- An emphasis on reliable communication as opposed to real-time communication, it is more important for packets to arrive *reliably* than to arrive *quickly*; and
- Very scarce processing and memory resources, as these sensors are often implemented on small low-power microprocessors.

This work provides overviews of routing protocols in ad hoc networks, swarm intelligence, and swarm intelligence applied to ad hoc routing. Various mechanisms that are commonly encountered in ad hoc routing are experimentally evaluated under situations as close to real-life as possible. Where possible, enhancements to the mechanisms are suggested and evaluated. Finally, a routing protocol suitable for such low-power sensor networks is defined and benchmarked in various scenarios against the Ad hoc On-Demand Distance Vector (AODV) algorithm.

Keywords: Ad hoc network, MANET, routing protocol, swarm intelligence, ant colony optimisation, ACO.

Supervisor: Prof. A.P. Engelbrecht
Department of Computer Science
Degree: Magister Scientiae

All meaningful knowledge is for the sake of action.
All meaningful action is for the sake of friendship.
-- John Macmurray

Acknowledgements

This work would not have existed without the following people:

- My father, Johann du Plessis, who has always been my greatest hero;
- My mentor, Prof. Andries Engelbrecht, who gave hours of academic support and crucial financial support;
- The people at B3 Solutions (Pty) Ltd where I gained most of the knowledge required for this work and who put up with my erratic schedule;
- My fabulous wife and best friend, Cynthia, who gives meaning to my life.

Bedankings

Hierdie werk sou nie tot stand gekom het sonder die volgende mense nie:

- My vader, Johann du Plessis, wat nog altyd my grootste held was;
- My mentor, Prof. Andries Engelbrecht, vir ure se akademiese en kritiese finansiële ondersteuning;
- My kollegas by B3 Solutions, waar ek meeste van die kennis nodig vir hierdie werk opgedoen het;
- My beeldskone vrou en beste vriend, Cynthia, wat betekenis aan my lewe gee.

□□

□□□□□□□□□□

- □□□□—Johann du Plessis, □□□□□□□□□
- □□□□—Andries Engelbrecht □□□□□□□□□□□□□□□□□□□□
- □□□□—B3 Solutions □□□□□□□□□□□□□□□□□□□□□
- □□□□□□□□□□□□—□□□□□□□□□□□□□□□□

Introduction

In an ad hoc network, mobile nodes communicate with each other using multi-hop wireless links. There is no stationary infrastructure to route packets. Instead, each node in the network also acts as a router, forwarding data packets for other nodes. Ad hoc networks are characterized by multihop wireless connections, frequently changing network topology and the need for efficient dynamic routing protocols. These networks have been studied in the past in relation to defense research, often under the name of packet radio networks [1, 2].

Recently there has been a renewed interest in this field due to the common availability of low-cost devices with radio interfaces [3]. Interest is also partly fueled by growing enthusiasm for running common network protocols in dynamic wireless environments without the requirement of specific infrastructures. A mobile ad hoc working group has also been formed within the Internet Engineering Task Force (IETF) to develop a routing framework for Internet Protocol (IP) based protocols in ad hoc networks [4].

The applications of ad hoc networks typically fall under two categories:

- Mobile Ad hoc NETworks (MANETS) such as mobile cellphones, laptops, Personal Digital Assistants (PDAs) etc, where the aim is to establish connectivity between devices [5]. The largest-scale example of this kind of network is probably the Tactical Internet (TI) network implemented by the US Army in 1997 [6]; and
- Ad hoc sensor networks encountered in many industrial applications, where the aim is to relay information from a collection of data gathering devices deployed over an area to a central point (and possibly back to the devices) [7, 8].

The issues present in typical wireless ad hoc networks are twofold. Firstly, ad hoc networks inherit the traditional problems of wireless networking and wireless communication [9]:

- The wireless channel does not have easily observable boundaries outside of which packet transmission is known to fail;
- The wireless channel is unprotected from outside interference;
- The wireless channel is significantly less reliable than a wired channel; and
- The channel has asymmetric and time-varying propagation properties.

Secondly, the multi-hop nature and the lack of fixed infrastructure add some issues that are specific to mobile ad hoc networks [10, 11]:

- Dynamic network topologies, presenting challenges in routing and link bandwidth allocation;
- Providing consistent communication to all devices subject to a changing environment;
- Conservation of power, which is essential to users of mobile wireless devices; and
- Global vs. local longevity, i.e., which "long-lived" routes are desirable.

Intelligent bandwidth allocation, power control and routing techniques are critical for these networks that have nodes with different data rate requirements, limited power and limited bandwidth. These techniques coordinate the nodes to relay information while exercising power control. They also let the network adapt to the removal and addition of different high and low rate communication sources, changing activity patterns, and incorporation of new services.

Recently, ant algorithms and swarm intelligence systems have been offered as a novel computational approach that replaces the traditional emphasis on control, preprogramming and centralization, with designs featuring autonomy, emergence and distributed functioning. These designs are proving flexible and robust, able to adapt quickly to changing environments and to continue functioning even when individual elements fail. The advantages of swarm intelligence are mainly due to the use of mobile agents and stigmergy [12-14]. These are:

- **Scalability:** The population of swarm agents can be adapted according to the network size. Scalability is also promoted by local and distributed agent interactions.
- **Fault tolerance:** Swarm intelligent processes do not rely on a centralized control mechanism. Therefore the loss of a few nodes or links does not result in catastrophic failure, but rather leads to graceful, scalable degradation.
- **Adaptation:** Agents can change, die or reproduce, according to network changes.
- **Speed:** Changes in the network can be propagated very fast, in contrast to many other algorithms.
- **Modularity:** Agents act independently of other network layers.
- **Autonomy:** Little or no human supervision is required.
- **Parallelism:** Agents' operations are inherently parallel.

These properties make swarm intelligence very attractive for ad hoc wireless networks. These algorithms also offer potential advantages for conventional infrastructured telecommunications, such as cellphone networks. The New Scientist [15] recently gave troubling details about problems with British Telecom's network, and the company's investigation of swarm intelligence as a potential solution. BT's 24 million users are coordinated through a conventional web controller that, in 1995, comprised of 30 programs with average memory requirements of 350 gigabytes. "Much of [the controller's]... time is spent just checking that all the elements of the network are working. It must also be constantly updated as new subscribers, new services, and new problems emerge. As it gets older it becomes harder to adapt, and a failure at the center could have potentially disastrous effects across the whole network" [15].

The distributed nature of swarm intelligence avoids the troubling bottlenecks that result from continuous use of such a centralised controller. The aim of this work is to apply these swarm intelligence techniques to sensor networks in the hope that the result will be a simple yet robust, scalable yet cost effective solution.

Some sensor networks employ very large numbers of nodes without unique node IDs [7]. When a user requests data from a specific area, the data from various nodes in that area are aggregated to form a complete representation. Other sensor networks require each node to have a unique ID, as for example in a parking-lot monitoring network where a unique ID is assigned to the sensor in each lot.

The focus of this work is only on ad hoc sensor networks where each node has a unique ID. These networks have a wide range of potential applications in tagging and tracking of assets and personnel, factories, plants and construction sites where operational and telemetry data have to be gathered (often called Supervisory Control and Data Acquisition or SCADA environments), and any other situation where it is more economical and/or convenient to use infrastructureless networking than to rely on centralised infrastructure [10, 16]. In these harsh environments, it is often necessary to seal the sensor nodes in rugged enclosures. As these nodes are difficult to reopen and often distributed over wide areas, they are required to function for extended periods (sometimes years) without battery replacement. The routing protocol on such a network must therefore be exceptionally power efficient.

This study aims to arrive at a routing protocol suitable for low-power sensor networks. The traditional ad hoc routing protocols are studied, including protocols that use swarm intelligence. Various common sub-mechanisms in these protocols are evaluated in conditions as close as possible to real-life. Some enhancements to the mechanisms are suggested and experimentally evaluated, and a new protocol is defined which includes these enhancements. This protocol, called the Ant Colony Optimisation Distance Vector (ACODV) protocol, is suitable for low-power sensor networks. Finally, ACODV is benchmarked in various scenarios against the AODV algorithm.

Objectives

The main objective of this thesis is to develop a routing protocol suitable for low-power, uniquely addressable ad hoc sensor networks. Towards this goal, the following sub-objectives are identified:

- To provide an overview of existing routing algorithms and mechanisms in mobile ad hoc networks.

- To provide an overview of swarm intelligence in general, and more specifically the ant colony optimisation metaheuristic.
- To establish an algorithm for routing in low-power ad hoc sensor networks.
- To establish a set of metrics used to measure the performance of the proposed routing algorithm, and then to use these metrics to compare the algorithm with established ad hoc routing algorithms.
- To investigate the factors influencing the performance of the proposed routing algorithm, and the algorithm's performance under various conditions.

Contribution

The main contributions of this work are:

- The testing under conditions approaching real-life of commonly used ad hoc routing mechanisms.
- The definition and categorisation of an extensive set of ad hoc routing metrics.
- The definition and experimental evaluation of a robust and power-efficient routing protocol, suitable for implementation on resource-constrained microcontrollers.

Additionally, an explicit goal of this work is to take a step towards standardising of ad hoc nomenclature. To this end:

- All ACO-related variables are named in accordance with the ACO metaheuristic as defined by Dorigo and Di Caro [17].
- A standardised set of performance metrics, as presented in previous works, is used to evaluate the performance of the algorithm.
- All control and data packets are named in accordance with the naming scheme used in the AODV algorithm.

Thesis Outline

Chapter 0 provides a brief taxonomy of ad hoc routing protocols. An overview of various proactive, reactive and hybrid protocols is given, and these classes of protocols are further subdivided into hierarchical and flat (i.e. non-hierarchical) protocols.

Chapter 2 introduces stigmergy and self-organisation. The chapter then moves on to introduce swarm intelligence (SI), which is based on stigmergy and self-organisation. A specific subcategory of SI called Ant Colony Optimisation (ACO) and its mathematical representation is then introduced.

Chapter 3 looks at ad hoc routing protocols that employ swarm intelligence. The protocols are again classified into proactive and reactive protocols. In each category, hierarchical and flat protocols are reviewed. The chapter ends with a review of some general swarm intelligence frameworks.

Chapter 4 lays the groundwork for meaningful evaluation of a protocol's performance by introducing and defining various performance metrics. The metrics are grouped into scenario metrics which describe the environment in which a protocol performs, and performance metrics which describe the protocol's performance in that environment. The chapter concludes with definitions of qualitative and quantitative protocol *features* which provide more insight into the protocol's operation.

In Chapter 5, sub mechanisms that are present in many routing algorithms are experimentally evaluated. The Route Request (RREQ) and Route Replay (RREP) mechanisms are tested, and the impact of some improvements evaluated. The influence of the backtracking mechanism on a protocol's performance is then examined, followed by experimental evaluation of parameters used by the ACO algorithm to make next-hop decisions.

Chapter 6 brings it all together by defining the operation of the ACODV protocol. The storing of routing information, configuration parameters and general operation of the protocol is described. The response of the protocol to varying node mobility and varying load conditions is experimentally evaluated and compared with AODV. The scalability of the two protocols is then evaluated in networks of up to 1500 nodes. Finally, functions not supported by ACODV are listed.

Chapter 7 concludes this work with a summary of the experimental results and future work resulting from this study.

A list of appendices provide a quick reference to ad hoc routing protocol acronyms, as well as a list of symbols used in this work.

Contents

List of Figures

List of Tables

Ad Hoc Network Routing Protocols

This chapter presents an overview of routing protocols used in ad hoc networks. An overview is given of the general routing problem, after which the criteria for the classification of ad hoc routing protocols are discussed. Using this classification, the various classes of routing protocols are presented with samples of each class.

1.1 Introduction

In an ad hoc network, each node functions simultaneously as a host and a router. Therefore, each node has to maintain some form of information regarding the network around it, and some algorithm governing the sending and receiving of data packets. This algorithm, together with the supporting information regarding network conditions, is called a routing protocol.

Routing protocols in ad hoc networks have to adapt quickly and elegantly to frequent, unpredictable changes in network topology, and they have to do so while conserving memory, power and bandwidth resources. When ad hoc networks are scaled up, they usually encounter excessive overhead in the routing messages, caused by the growing number of nodes and amplified by higher node mobility. Growing networks also lead to excessive routing table sizes, which typically have to be broadcasted to other nodes – again causing network overhead.

Most routing protocols for ad hoc networks use a derivative of either link-state or distance-vector routing [18]. In link-state routing each node maintains a picture of the global network topology, usually through periodical flooding of routing table information to its neighbours. When a node receives an update packet, it updates its view of the network and the link-state to other nodes by applying a shortest-path algorithm to choose the best next-hop node for each potential destination node in the network. Therefore higher node mobility will require more frequent flooding, a very undesirable characteristic. In a network with n nodes, the link-state routing overhead O will be a function of n^2, or otherwise stated $O(n^2)$ [19]. In larger networks, and especially under higher mobility, routing overhead will quickly consume all available power and bandwidth resources, making these protocols unsuitable for power and bandwidth restricted ad hoc networks. In distance-vector routing on the other hand, each node only maintains a table with the cost (in terms of power, number of hops, or any number of factors), and the direction (the next downstream node) needed to send a packet to any node. In many distance-vector routing algorithms these vectors are updated implicitly by data packets as they travel through the network – hence fewer special overhead packets have to be sent. In stead, each time a packet it sent from one node to another, the packet is augmented with local information regarding the sending node. When receiving a packet, the receiving node updates its distance vector tables with information from the packet, replaces the local information with the local node’s own local information, and sends the packet to the next node. Although distance-vector protocols typically require much less routing overhead than link-state protocols, they suffer from other drawbacks such as slow convergence to near-optimal routes, and a very troubling tendency to create transmission loops (sending the packet around and around the same circle of nodes) [20].

Some applications overcome the scalability issue by dividing the network into clusters [21, 22]. Clustered nodes only communicate directly with other nodes in their cluster, and any packets for outside nodes are sent to a designated cluster head node. The nodes therefore form a layered or hierarchical network structure. If node mobility is low, hierarchical networks can offer a very attractive solution to scalability. However, frequent topology changes will mean frequent changes to the clusters. In high mobility networks, the updates to the clusters can become the major network overhead both in terms of computational power and bandwidth. Additionally, clustered networks imply that some nodes will inevitably carry more load than others. In homogenous ad hoc networks the network load has to be spread as evenly as possible, making a hierarchical network infeasible. An additional drawback of hierarchical networks is that it sometimes introduces a critical reliance on specific nodes – if the clusterhead nodes fail, all the nodes in the cluster may be out of reach. Flat routing schemes, on the other hand, offer more node redundancy but at increased processing or bandwidth cost.

The rest of this chapter is organized as follows. Section 1.2 introduces the criteria used to classify ad hoc routing protocols. Sections 1.3, 1.4 and 1.5 present *proactive, reactive* and *hybrid* hierarchical routing protocols respectively. Sections 1.6, 1.7 and 1.8 present *proactive, reactive* and *hybrid* flat routing protocols respectively. The protocols discussed in these sections are not intended to be exhaustive, but to represent an overview of the different paradigms used. Section 1.9 concludes this chapter.

1.2 Classification of Ad Hoc Routing Protocols

The primary axis used for classification of ad hoc routing protocols is usually one of the following [19, 23, 24]:

- Whether the protocol forms a hierarchical or flat (i.e. non-hierarchical) network structure; or
- Whether the protocol is proactive (also called global or table-driven) or reactive (also called on-demand or source-driven).

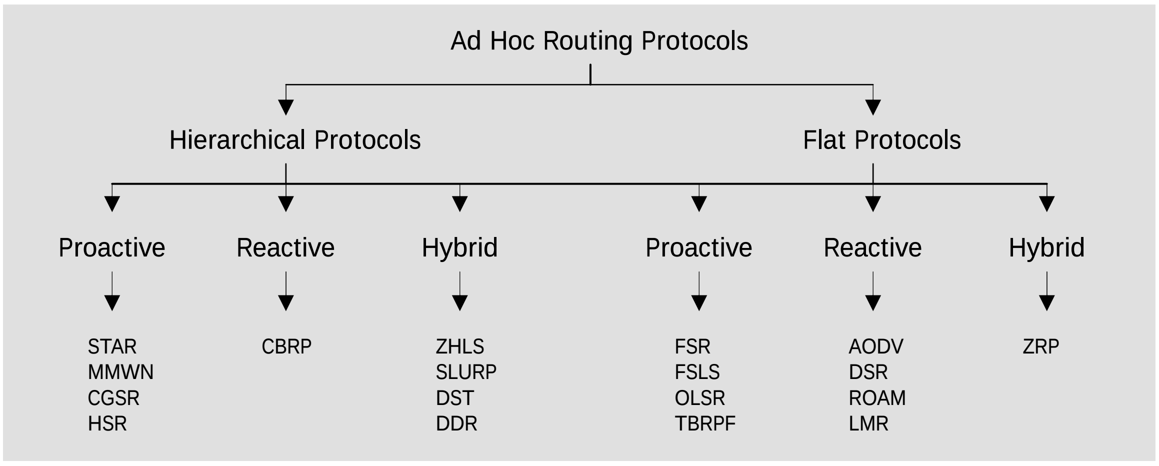


Figure 1: Taxonomy of ad hoc routing protocols

Figure 1 presents a hierarchical overview of ad hoc routing protocols (the abbreviations used in Figure 1 are defined in Appendix A). It has to be noted that there are many different axii along which to classify ad hoc routing protocols. Lang maintains that a classification along the hierarchical or flat network structure axis is not valid, since a “flat” network structure is not really a property in itself, but rather the absence of a hierarchical property [25]. However, since some networks generally lend themselves better to hierarchical networks and *vice versa*, a network designer may find a classification along this axis useful. This work therefore accepts a classification along the hierarchical or flat network structure axis on the basis that it yields *useful information*, even though such a classification can be argued to be logically incorrect.

Proactive routing protocols attempt to maintain at all times the information necessary to route information to any node in the network, and are usually a derivative of the link state algorithm. Reactive protocols only acquire (or update) the routes on demand, and are usually a derivative of the distance vector algorithm. Since some protocols are a hybrid of proactive and reactive paradigms, a classification of protocols using this criterion also requires a third “hybrid” category.

This thesis classifies ad hoc routing protocols primarily as hierarchical or flat, and secondarily as proactive, reactive or hybrid (see Figure 1). A more comprehensive framework for the taxonomy of ad hoc routing protocols is presented in [26].

1.3 Proactive Hierarchical Protocols

Proactive protocols update routing information continuously to ensure that active routes to all possible destination nodes are continuously maintained. The routing information is usually kept in a number of tables, and tables are updated either periodically or when nodes become aware of changes in the network topology. To reduce the amount of network overhead, this class of protocols partitions the network into various clusters. Nodes usually keep direct routing information for sending packets to other nodes in the same cluster, while packets for nodes in other clusters are routed through designated clusterhead nodes. This class of protocols is usually suited for networks with relatively low mobility, as both the proactive routing scheme and the clustering mechanism require significant network overhead with higher mobility.

1.3.1 Source-Tree Adaptive Routing (STAR)

In the STAR protocol [27], each node maintains a *source tree* – a set of preferred links to each destination node. STAR is therefore a derivative of the link state algorithm. STAR uses a *least overhead routing approach* (LORA) to significantly reduce the amount of routing overhead required. The STAR algorithm also supports an approach making the updating of routing information conditional – called the *optimum routing approach* (ORA). As a result of the significantly reduced routing overhead, STAR scales well in large networks. However as STAR requires each node to maintain at least a partial topology of the network, this protocol has large memory and processing requirements in large and/or highly mobile networks. The topology that each node has to maintain is determined by the source tree reported by the node's neighbours, and this may keep changing as the nodes report different source trees.

- **Advantages:** Use of LORA and reduced routing overhead.
- **Disadvantages:** Large memory and processing overhead.

1.3.2 Multimedia Support in Wireless Networks (MMWN)

In MMWN [28] each node in the clustered hierarchy is designated as either a *switch* or *endpoint* node. Endpoint nodes are nodes that are within one hop of a switch node (similar to cellphones connecting to cellphone base towers), and traffic originates and ends at endpoint nodes. Switch nodes relay information to other switch nodes and ultimately to endpoint nodes. Switch nodes may have an arbitrary wireless hop distance between them (see Figure 2). The location management for each cluster is done by one node designated as a *location manager* (LM), and all topological information is stored in a dynamically distributed database. Routing overhead in MMWN is significantly reduced by the fact that only the LMs perform location updating and finding. However, the location updating and finding messages have to travel through the hierarchical tree of the LMs, making the location finding and updating very complex. Additionally, changes in the hierarchical cluster *membership* of LMs will affect the hierarchical management tree. This causes complex consistency management problems in the implementation of MMWN.

- **Advantages:** Reduced overhead which can be even more reduced by using LORA.
- **Disadvantages:** Complex mobility management and cluster maintenance overhead which is amplified by consistency problems in management of LMs. Failure of clusterhead or gateway nodes can be catastrophic to the network.

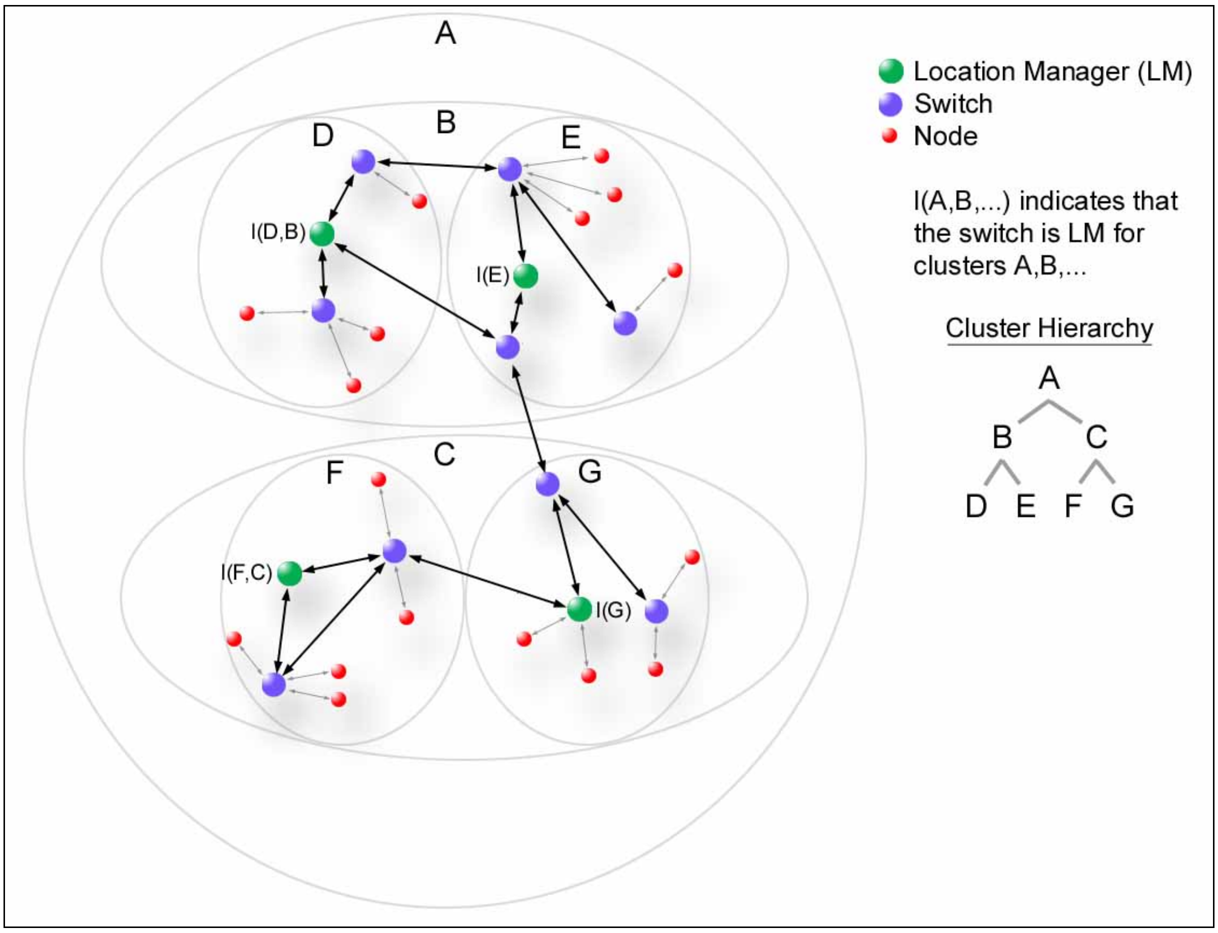


Figure 2: MMWN routing using a clustering hierarchy

1.3.3 Clusterhead Gateway Switch Routing (CGSR)

In CGSR [21], a *Least Clusterhead Change* (LCG) clustering algorithm is used to group the network into clusters, and a clusterhead node is selected in each cluster for inter-cluster communications. CGSR allows nodes to belong to more than one cluster at a time – these nodes facilitate inter-cluster communications and are called gateway nodes. CGSR therefore replaces the multi-layered hierarchy of MMWN with multiple node functions. Inter-cluster routes in CGSR are in the format (see Figure 3):

SourceNode→Clusterhead→Gateway→Clusterhead→Gateway→ …→DestNode

Each node in CGSR maintains two tables – a routing table with the distance vectors to other nodes, and a cluster membership table. Nodes periodically broadcast their cluster membership tables to neighbouring nodes – which may cause significant bandwidth overhead in highly mobile networks. The cluster membership table also contains the clusterhead of each node in the table. Since packets are routed through the clusterheads, the routing table of a node only maintains the direct routes to nodes inside its cluster. Outside the node's cluster, the node only maintains routes to the clusterheads. When a new packet arrives, the node first finds the destination's clusterhead in the cluster membership table, and then routes the packet to that node. The specified clusterhead will then route the packet to the destination node.

- **Advantages:** Greatly reduced routing table sizes compared to other distance vector protocols.
- **Disadvantages:** Maintaining clusters may take a lot of computational and bandwidth overhead. Failure of clusterhead or gateway nodes can be catastrophic to the network.

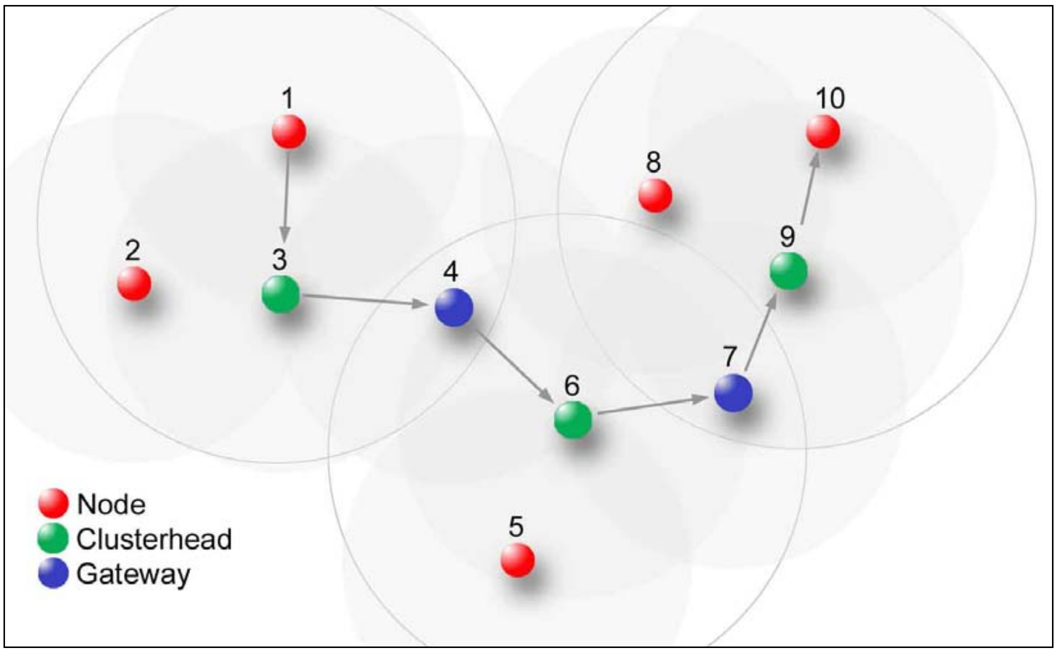


Figure 3: Routing through clusterheads and gateways in CGSR

1.3.4 Hierarchical State Routing (HSR)

HSR [29] divides the network into multiple layers by using a recursive clustering scheme. To create layers, nodes selected as clusterheads in lower layers are clustered together into the next (higher) layer. HSR also uses the three types of nodes used in CGSR – clusterheads, gateways and internal cluster nodes. The nodes in higher layers send link state information only to other nodes in the same layer, thereby greatly reducing routing overhead. The clusterhead nodes then summarise this information and floods the summarised information to the lower-level nodes in its cluster. HSR therefore creates "virtual" or "tunnel" links between nodes in the network. A packet originating at a lower-level node is sent "up" the hierarchy to the first level where the destination node is known, and then "down" that hierarchy to the specified destination. HSR uses a *hierarchical ID* (HID) for each node. The HID is defined as the sequence of MAC addresses of the nodes from the top hierarchy to the node itself. On receiving route updates from a higher level node, each node can update its HID (and therefore its cluster membership) dynamically. Gateway nodes can communicate with multiple clusterheads, allowing them to be reached via multiple paths. This implies that gateway nodes have multiple HIDs – which is similar to routers in the Internet. HSR uses *home agents* to separate the mobility management from the physical hierarchy.

- **Advantages:** Greatly reduced routing overhead, multiple paths to gateway nodes.
- **Disadvantages:** HIDs are longer than conventional IDs, creating a slight increase in bandwidth consumption. Continuously changing HIDs can make it difficult to keep track of nodes. Failure of clusterhead or gateway nodes can be catastrophic to the network.

1.4 Reactive Hierarchical Protocols

Reactive hierarchical protocols partitions the network into clusters to reduce the amount of overhead required if each node has to keep explicit routing information to each other node. Nodes usually keep direct routing information for sending packets to other nodes in the same cluster, while packets for nodes in other clusters are routed through designated clusterhead nodes. This class of routing protocols aims to reduce routing overhead by only acquiring routes when they are needed. This eliminates the continuous drain of overhead incurred by proactive routing protocols, but also introduces a longer delay before packets are delivered.

1.4.1 Cluster-Based Routing Protocol (CBRP)

CBRP [30] divides the network into a hierarchy similar to HSR. The biggest difference between CBRP and HSR is that CBRP uses distance vector routing compared to HSR's link state routing. In CBRP, only clusterhead nodes exchange routing information and co-ordinate routing of packets inside their clusters. The routing overhead in CBRP, already lowered by using a hierarchical topology, is further lowered by the distance vector routing mechanism – routing overhead is largely concerned with cluster maintenance. The biggest drawback of this protocol is that topology changes take long to be propagated through the network, causing nodes to have inconsistent topology information, which in turn may cause frequent temporary routing loops.

- **Advantages:** Very low routing overhead.
- **Disadvantages:** Temporary routing loops, cluster maintenance overhead.

1.5 Proactive/Reactive Hybrid Hierarchical Protocols

This class of routing protocols aims to combine the best properties of proactive and reactive protocols. The network is partitioned into clusters, and different routing schemes are used for inter- and intra cluster routing. A proactive protocol that provides quick delivery of packets (but reduced scalability due to increased network overhead) is usually used to deliver packets to nodes in the same cluster, while a reactive protocol that provides reduced overhead (but increased packet delay) is usually used to deliver packets to nodes in other clusters.

1.5.1 Zone-based Hierarchical Link State (ZHLS)

In ZHLS [31] the network is divided into non-overlapping zones. ZHLS requires each node to be aware of its position (by using a GPS or similar device), and assigns both a Node ID and a Zone ID to each node. This solves many of the inefficiencies caused by overlapping zones as encountered in the Zone Routing Protocol (ZRP is discussed in section 1.8.1). The ZHLS network hierarchy has two levels – the top zone level, and the bottom node level. For intra-zone communications a proactive protocol is usually used, though any proactive or reactive protocol can be employed. When a route to a destination outside the source node's zone is required, the source node broadcasts a zone-level route request to all other *zones* – which significantly reduces the overhead compared to a protocol that uses flooding to all other *nodes*. Location management in ZHLS is greatly reduced by the use of positioning devices, and no clusterhead or location manager is needed to coordinate routing. This reduces the processing *and* bandwidth requirements of ZHLS compared to both proactive hierarchical protocols such as CGSR, HSR and MMWN, and reactive protocols such as AODV and DSR. However ZHLS requires each node to have a pre-programmed static zone map, which may not be feasible for many applications where the geographical boundary of the network is dynamic. For networks where the positioning device and zone map requirement can be met, ZHLS is one of the most highly adaptive, scalable protocols available.

- **Advantages:** Very low routing overhead and low processing requirements, no critical nodes, fast route discovery, very scalable.
- **Disadvantages:** Requires a positioning device for each node and a static zone map.

1.5.2 Scalable Location Update Routing Protocol (SLURP)

SLURP [32] organizes the network into non-overlapping zones similar to ZHLS, and also requires each node to be equipped with a positioning device. SLURP further reduces the overhead of route maintenance by introducing a *home region* for each node. The home region of each node is determined using a static, globally known, many-to-one mapping function in the form $f(NodeID) \rightarrow RegionID$. An example of such a mapping function is:

$$f(NodeID) = g(NodeID) \cdot mod(k)$$

where:

$g(NodeID)$	=	*A random number generating function using NodeID as seed.*
k	=	*The total number of home regions in the network.*

Since the *NodeID* of each node is constant, the function will always calculate the same home region for any given node. This allows any node to calculate the home region of a destination node given the node's ID. When a node leaves one zone and enters another, it unicasts a location update message to its home region. Any node in the home region that receives the unicast will broadcast it to all other members of the home region. Therefore any node that wants a route to a destination node can unicast a route request packet to the destination node's home region. Any node in the home region receiving this request will return the destination node's current zone location. Once the node's current zone is known, packets are sent to the zone using a geographical forwarding algorithm such as *most forward with fixed radius* (MFR) [33] or *selection diversity forwarding* (SDF) [34]. Once the packet reaches the destination's zone it is usually forwarded to the destination node using a source routing protocol such as DSR (DSR is described in section 1.7.3), although other link state protocols could also be used.

- **Advantages:** Faster route discovery and less route maintenance overhead using home regions.
- **Disadvantages:** Requires a positioning device for each node and a static zone map.

1.5.3 Distributed Dynamic Routing (DDR)

Although DDR [35] uses spanning trees to route packets (similar to *distributed spanning trees based routing protocol* (DST) [36]) DDR does not require the critical root nodes used in DST. In stead, trees are constructed by beaconing messages which are sent periodically by all nodes to their neighbouring nodes (i.e., nodes which are exactly one hop away). The network forms a "forest" of spanning trees. Trees are connected with *gateway nodes* – defined as nodes which are within transmission range of a node, but belong to a different spanning tree. The protocol uses a zone naming algorithm to assign a unique zone ID to each tree. The network is therefore clustered into different non-overlapping *trees* or *zones*, the two terms being used interchangeably. The DDR routing algorithm is initialized in 6 phases, namely:

- **Preferred Neighbour (PN) Selection:** During this phase each node selects a preferred neighbour. The preferred neighbour of a node is the neighbouring node that has the highest number of neighbours (the "most connected" node, see Figure 4). If more than one node has the same highest number of neighbours, the node with the highest node ID will be selected. The way in which a node is selected therefore forms a monotonic increasing function depending on its number of neighbours and its ID number.
- **Forest Construction:** After preferred neighbour selection, a forest is constructed by connecting each node to its preferred neighbour. Nikaein *et al* provide mathematical proof that, regardless of the network topology, this approach always yields a "forest" (i.e., a graph without cycles).
- **Intra-tree Clustering:** An intra-tree clustering algorithm is executed to determine the structure of each tree or zone.
- **Inter-tree Clustering:** After zones are formed, beacon messages are sent to determine which nodes are within transmission range of a node that belongs to a different zone. This information is used to determine connectivity between the zones.
- **Zone Naming:** The zone naming algorithm is executed to name each zone.

- **Zone Partitioning:** After zones are named, the network is partitioned into non-overlapping zones.

The timing and execution of these 6 phases are done by beaconing messages periodically exchanged by nodes. After the network has been initialized, messages are routed using a *hybrid ad hoc routing protocol* (HARP) [37]. HARP uses the intra-zone and inter-zone tables created by the DDR algorithm to determine paths between source and destination nodes. One of the drawbacks of DDR is that routes with many neighbours will be selected as preferred neighbours by multiple nodes, and may become bottlenecks. This can cause resource contention for the preferred nodes, and can also cause the more preferred nodes to fail quicker as their power is drained. Both these effects can have a significant impact on the DDR system throughput.

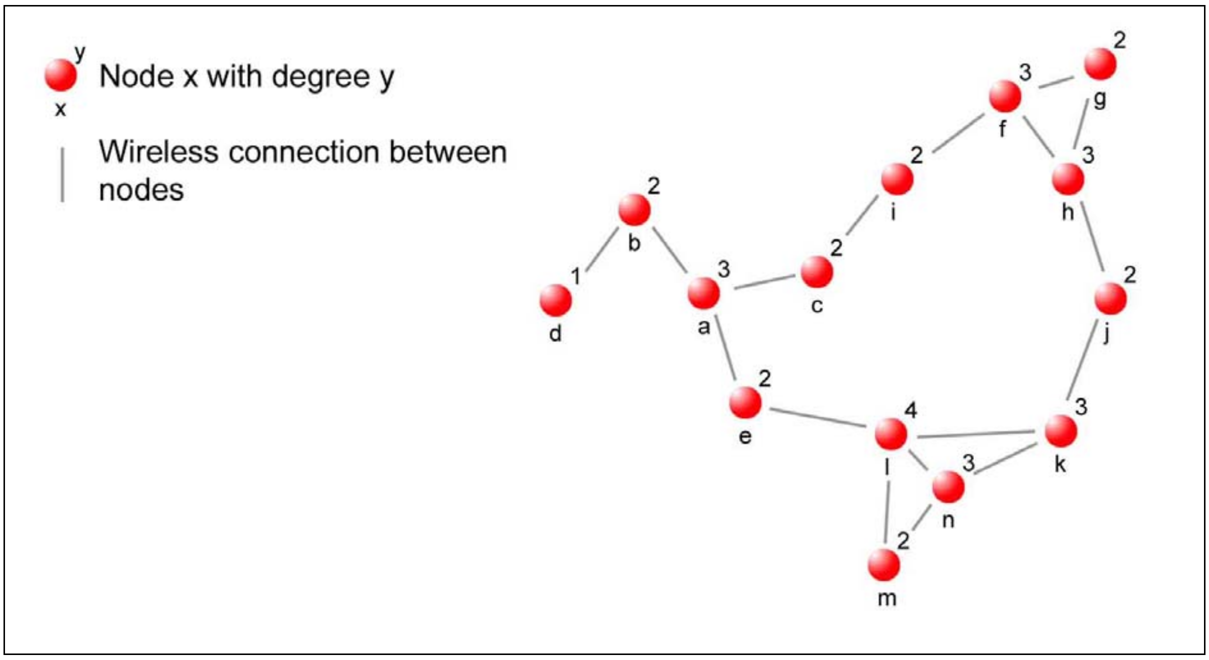


Figure 4: Degree of connectivity in DDR

- **Advantages:** DDR does not need a static zone map like ZHLS, and DDR does not need critical root or clusterhead nodes.
- **Disadvantages:** DDR could have high resource contention around preferred nodes, and preferred nodes could get drained of power quicker than others.

1.6 Proactive Flat Protocols

Proactive flat protocols update routing information continuously to ensure that active routes to all possible destination nodes are maintained. This ensures quick delivery of packets, but at the cost of higher network overhead. This class of protocols is less scalable than hierarchical protocols, as each node has to keep explicit routing information to each other node.

However nodes may have higher mobility than in clustered networks, as the network overhead of maintaining clusters is eliminated.

1.6.1 Fisheye State Routing (FSR)

FSR [38] is a very simple and efficient protocol based on the link state algorithm. Each node in the FSR network maintains a complete topology map of the network, and sends periodic link state updates to other nodes. The difference between FSR and conventional link state routing is the frequency at which updates are sent, and the information contained in the updates. Link state updates in FSR are periodic in stead of event-triggered, and the frequency at which nodes sent updates is inversely proportional to the distance (hop-count) between the source and destination nodes. Nodes therefore frequently exchange updates with their neighbours, and less frequently with further away nodes. Furthermore the link state updates of neighbours within a certain hop-count (called the *scope*) are always included in update packets, while the information of nodes outside the scope is included less frequently (see Figure 5). The result is that nodes have up-to-date information regarding nodes close to them, and more outdated information for farther away nodes. The result is a significant reduction in the amount of link state information exchanged.

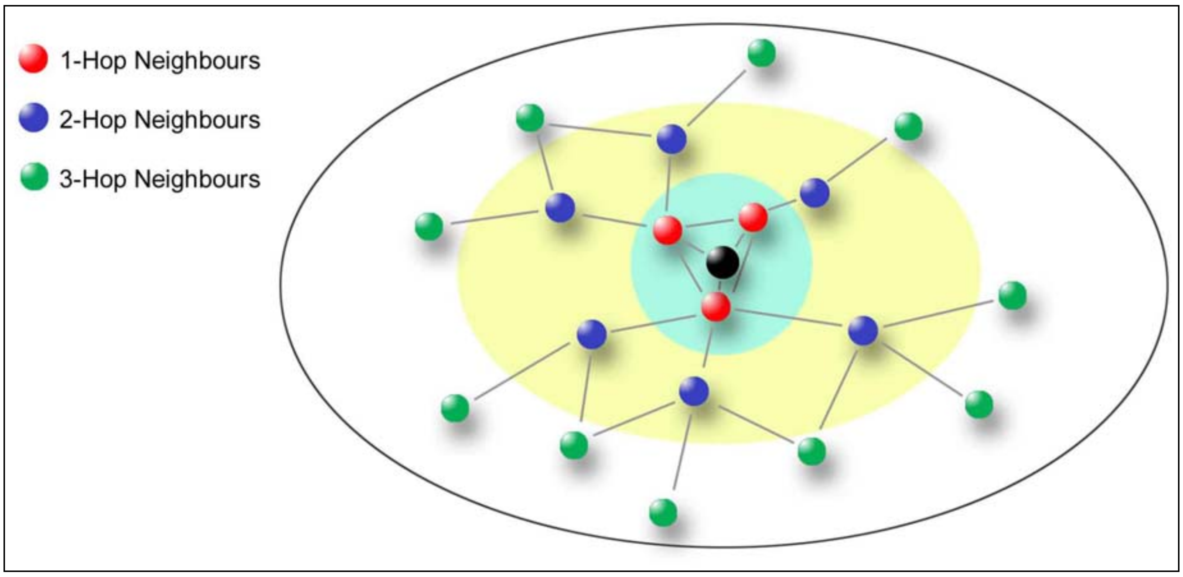


Figure 5: Scope of the FSR "fisheye"

Since the information for any further away node is outdated, the source node will only send the packet in the general direction of the direction node. However, as the packet gets closer to the destination node, the routing information in the nodes become progressively better, thereby compensating for the inaccurate information contained in the farther away source nodes.

Fuzzy Sighted Link State (FSLS) is a similar routing algorithm, introducing an exponential function to determine how often update packets should be sent. FSLS and “myopic” routing algorithms are discussed in [39].

- **Advantages:** FSR has reduced route maintenance overhead, and route accuracy can be traded off against bandwidth consumption by tuning the frequency of updates.
- **Disadvantages:** The accuracy and delay times can be very poor in highly mobile networks.

1.6.2 Optimised Link State Routing (OLSR)

OLSR [40] is based on the link state algorithm, but reduces routing overhead with the use of a *multipoint relaying* (MPR) strategy. When a node wants to send topology updates, it selects a group of neighbouring nodes to retransmit the routing packets – these nodes are called the multipoint relays of the source node. If a node receives a topology update packet from a node for which it is *not* a multipoint relay, it will update its topology with the information in the packet but will not rebroadcast the packet. To select multipoint relay nodes, each node follows the following algorithm (see Figure 6):

1. Each node periodically transmits a “hello” message which contains a complete list of the node’s one-hop neighbours.
2. When a node receives such a hello message, it compares the one-hop neighbours of the message with its own list of one-hop neighbours. Any node in the hello message but not in the node’s own list is a two-hop neighbour.
3. The node selects the smallest subset of one-hop neighbours necessary to reach all two-hop neighbours as its multipoint relay nodes.

In this way, each node determines a route that is optimal in terms of hop-count to every known destination in the network, and greatly reduces network routing overhead since not all nodes forward routing messages. OLSR also reduces the *size* of routing packets by letting a node only send routing updates for nodes that selected the node as a multipoint relay. This means that a node can only be reached through its multipoint relay nodes.

When a packet has to be sent to a destination node, OLSR calculates the shortest path to the node using the topology information in its routing tables. Because OLSR significantly reduces the number of broadcast retransmissions, this algorithm is most effective in networks with dense node distribution and frequent communication. When the network is more scarcely populated, every node becomes a multipoint relay and the advantage of using OLSR is lost.

- **Advantages:** OLSR uses reduced network overhead and reduces resource contention.
- **Disadvantages:** The reduced network overhead advantage of OLSR is slightly offset by the increase in network overhead due to frequent 2-hop hello messages.

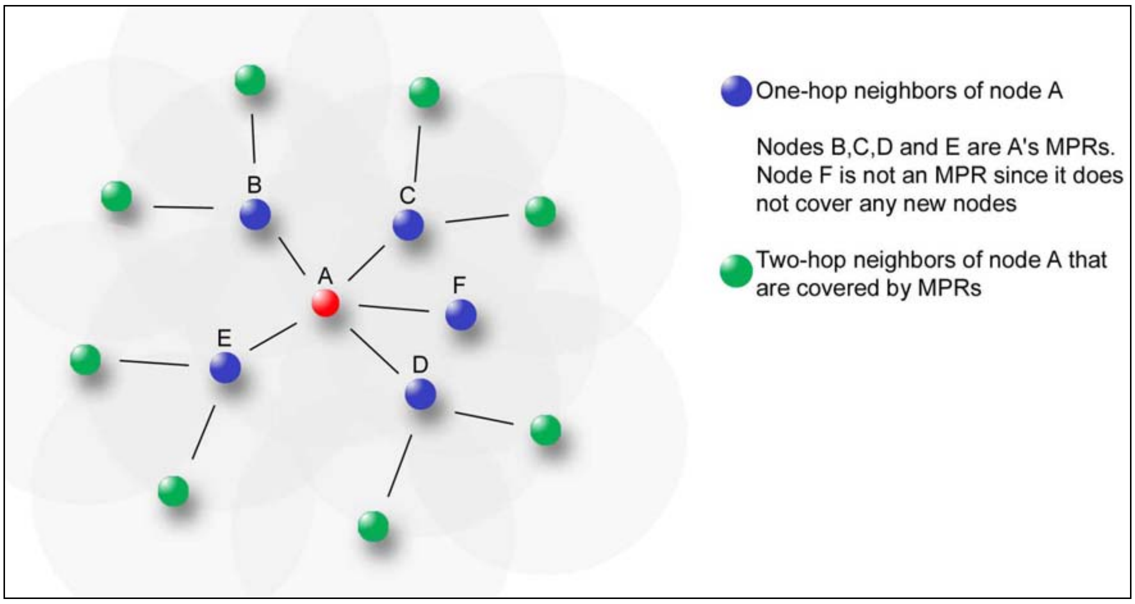

A
B
C
D
E
F
One-hop neighbors of node A
Nodes B,C,D and E are A's MPRs. Node F is not an MPR since it does not cover any new nodes
Two-hop neighbors of node A that are covered by MPRs

Figure 6: Selection of MPRs in OLSR

1.6.3 Topology Broadcast Reverse Path Forwarding (TBRPF)

TBRPF [41] is another proactive link state protocol. It is based on an earlier algorithm called *Extended Reverse Path Forwarding* (ERPF) which is described in [42]. The TBRPF protocol consists of a *neighbour discovery module* and a *routing module*. Neighbour discovery is done by each node periodically sending "hello" messages. These messages are differential, i.e. they contain only the relative changes (nodes that were added or removed) in the neighbours of the source node. Each node maintains a spanning tree for each of its neighbour nodes, using a modified version of Dijkstra's algorithm [43] to build the tree. Each node also maintains a list of parent nodes for each other node in the network, along with a topology table including the cost and sequence number of each other node in the network (that is, each other node that this node is aware of). TBRPF can operate in *partial topology mode* where each node only keeps a partial network topology and uses an algorithm similar to OLSR to decide which topology updates to send to other nodes; or *full topology mode* which keeps a full network topology at each node and thus provides extra robustness at the cost of more network overhead.

TBRPF introduced two major improvements over OLSR, namely that it created spanning trees of arbitrary distance (compared to a two-hop limit in OLSR) and it reduces overhead by sending differential topology updates. Another improvement is that TBRPF can use arbitrary link metrics to compute the spanning trees (if the links are symmetric) while OLSR only uses the number of hops.

- **Advantages:** Greatly reduced network overhead and the ability to use a combination of metrics to decide on the best route.
- **Disadvantages:** High memory and computational overhead.

1.6.4 Destination-Sequenced Distance Vector Protocol (DSDV)

DSDV [44] is a modification of the basic distance vector routing algorithm that guarantees loop-free paths by adding a sequence number to each packet sent from a source node. Each node in the network keeps a record of the source node, destination node and sequence number of each packet it receives. If it receives the same packet again, the packet is discarded. Two types of network update packets are used – called *full dump* and *incremental* packets. Full dump packets contain the entire network topology known to the source node, and incremental packets contain only the changes in the network topology. Incremental packets are sent more frequently than full packets to reduce network overhead. Although this scheme reduces bandwidth overhead, the overhead is still significant and this protocol will not scale well to large networks.

- **Advantages:** DSDV guarantees loop-free transmissions.
- **Disadvantages:** DSDV has significant network overhead and only uses hop-count as route metric.

1.6.5 Wireless Routing Protocol (WRP)

WRP [45] is an extension of the DSDV protocol. It introduced two improvements over DSDV:

1. In addition to the sequence number history that each node keeps, each node also keeps a record of the node *from which* the packet was received. This predecessor information helps to prevent temporary transmission loops, and
2. WRP provides for an arbitrary route metric to be used.

WRP is referred to in many other works, mainly due to the fact that it is one of the earliest proposed ad hoc routing algorithms. Murthy and Garcia-Luna-Aceves later continued the work, which lead to the WRP-lite protocol [46] and later the *Bandwidth Efficient Source Tracing* (BEST) protocol [47].

- **Advantages:** WRP can use an arbitrary route metric and prevents both transmission- and temporary loops.
- **Disadvantages:** WRP consumes more memory and computational resources than DSDV.

1.6.6 Distance Routing Effect Algorithm for Mobility (DREAM)

DREAM [48] requires each node to have a positioning device, and nodes periodically exchange positional information. Exchanging positional information has the advantage that it consumes significantly less bandwidth than exchanging complete link state or distance vector information. Routing overhead is further reduced by using a phenomenon called the *distance effect* – two nodes appear to be moving slower with respect to each other with increasing distance. DREAM therefore updates location information for distant nodes less frequently and less accurately than for nearby nodes, similar to FSR (FSR is discussed in section 1.6.1). This is done by limiting a packet's lifetime in terms of the packet's hop-count – after the number of hops specified in the header of the packet, it will be deleted. Packets with long lifetimes (which will reach faraway nodes) are sent less frequently than packets with short lifetimes. DREAM is a proactive protocol since it continuously updates positional information (i.e. not on-demand). Since nodes are aware of their own positions, they are also aware of their own speed. Nodes that move at higher speeds transmit positional information more frequently than slow moving nodes.

When a node has to send a packet, the node forwards the packet in the general direction of the destination node. Consecutive nodes should have increasingly accurate information regarding the node's position, and will forward the packet to the destination node. Basagni *et al* claim that this scheme makes the protocol inherently loop-free, since the packets travel away from the source node. This claim may not be true in highly mobile networks. A similar approach is followed in *location aided routing* (LAR), which attempts to predict a *target zone* where the node is likely to be using previous location and speed information, and then forwards packets to that zone (LAR is not discussed in this work, the interested reader is referred to [49]).

The bandwidth-saving measures used in DREAM are very effective at conserving network resources, making DREAM a highly scalable protocol.

- **Advantages:** DREAM scales very well to large networks and has low bandwidth and computational overhead.
- **Disadvantages:** Each node is required to have a positioning device.

1.7 Reactive Flat Protocols

This class of protocols does not partition the network into clusters, and routes are only acquired when they are needed. This makes this class of protocols suitable for highly mobile networks, as the network overhead of maintaining both routes and clusters continuously are eliminated. However, this comes at the price of lower scalability and possibly increased packet delay as packets have to wait for routes to be acquired. Nodes that do not originate or forward packets create almost no network traffic, which makes this class of protocols also suitable for networks with infrequent communication bursts.

1.7.1 Ad Hoc On-Demand Distance Vector Routing (AODV)

In conventional distance vector routing [1, 50], each router maintains a routing table giving the hop-count from itself to all possible destinations. A router periodically broadcasts its routing information to each of its neighbour routers, and uses the values received from neighbour routers to compute updated values for its own table. By comparing the distances received for each destination from each of its neighbours, a router can determine which of its neighbours has the shortest path toward each destination. When a router receives a packet for forwarding to some destination, the router simply forwards the packet to the correct next hop router. By transmitting routing table updates when any information in the table changes (such as a node being added or removed) the algorithm converges more quickly to the correct path, but the overhead in CPU time and network bandwidth for transmitting routing updates increases. Examples of distance vector routing protocols include the routing protocol used in the DARPA Packet Radio Network and the original routing protocol for the ARPANET. AODV extends traditional distance vector routing to an ad hoc environment.

AODV [51] is one of the most discussed and most mature ad hoc protocols currently available, and is an important part of the IETF MANET working group's work. It is an extension of DSDV (DSDV is described in section 1.6.4) with the same destination sequence and beacon message mechanisms. The major difference is that in AODV *all* exchanges of routing information is initiated when a packet has to be sent – no periodic transmissions are done (except for optional periodical one-hop hello messages).

When a source node in AODV has to send a packet to a destination node for which it does not already have a route, it initiates a *route discovery* process. This is done by broadcasting a *route request* (RREQ) packet to all its neighbours. If a neighbour does not have a valid route, the neighbour re-broadcasts the packet to other nodes in the network. The destination sequence mechanism ensures that these multicasts are propagated through the network loop-free. During the route request process, nodes that receive RREQ packets record in their routing tables the node from which the RREQ packet was received, thereby setting up a reverse path back to the source node. RREQ packets can also be sent using an *expanding ring search*. In this scheme, each RREQ packet has a fixed time-to-live (TTL) expressed in number of hops. If a search with a small TTL fails, a search with a larger TTL will be initiated until the destination is found. This reduces network traffic generated by a route request, but also significantly increases the delay associated with route discovery.

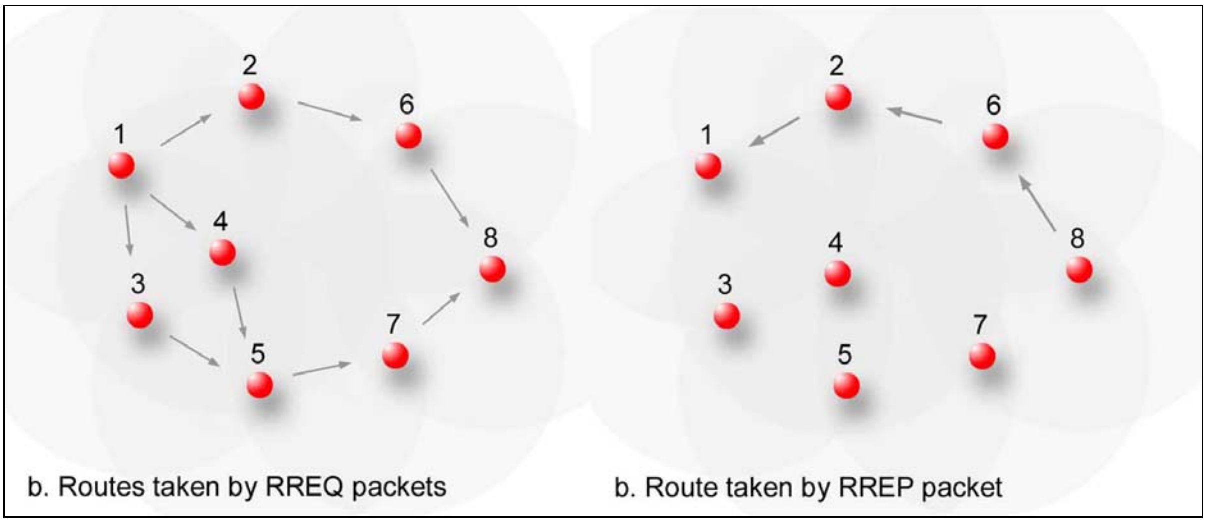

1
2
3
4
5
6
7
8
b. Routes taken by RREQ packets
1
2
3
4
5
6
7
8
b. Route taken by RREP packet

Figure 7: Routing of AODV RREQ/RREP packets

At some point, the RREQ packets should reach either a node with a valid route to the destination, or the destination itself. This node (whether a node with a valid route or the destination itself) will then unicast a *route reply* (RREP) packet back to the node from which it received the RREQ packet (see Figure 7). As the RREP packet is routed back to the source node, nodes along the route will record in their routing tables the node from which the RREP packet was received, thereby setting up the forward path back to the destination node. A route timer is associated with each routing table entry, causing the entry to be deleted when it expires. Since routes are set up in the reverse direction, standard AODV only supports symmetrical links.

Packets in AODV have to be acknowledged on a node-to-node (as opposed to end-to-end) basis. If an attempt to send a packet to the next downstream node in a route fails (i.e. no acknowledgement was received), the node will send a *route failure notification* (RERR packet) to each upstream node in the route, notifying them of the failure. The upstream nodes will then erase that part of the route from their routing tables. Once the route RERR packet reaches the source node, it can choose to initiate another RREQ if needed. Alternatively each one of the upstream nodes can initiate a RREQ to attempt to find a new route to the destination, and only send the route RERR packet upstream if their attempt fails.

- **Advantages:** AODV has low memory and computational consumption in nodes, and very low network bandwidth overhead.
- **Disadvantages:** Because AODV does not allow for multipath routing, new routes always have to be discovered on route failure [52].

1.7.2 Ad Hoc On-Demand Multipath Distance Vector Routing (AOMDV)

An interesting extension to AODV is introduced by Marina and Das in [53]. The Ad hoc On-demand Multipath Distance Vector (AOMDV) routing protocol extends AODV to a multipath routing protocol. When a protocol attempts to find multiple routes to a destination, it is often useful to have a measure of the similarity of routes. The following classification of routes according to the routes' similarity can be made [54]:

- **Node Disjoint** or totally disjoint routes have no links or nodes in common except the source and destination,
- **Link Disjoint** routes have no links in common, but the routes may have nodes in common, and
- **Non-disjoint** routes may have links and nodes in common.

The AOMDV protocol creates link disjoint paths by using a property of flooding, which Marina and Das describe. Marina and Das describe the property and its proof as follows [53]:

Property 1. Let a node $\boldsymbol{S}$ flood a packet m in the network. The set of copies of m received at any node $\boldsymbol{I}$ ($\neq \boldsymbol{S}$), each arriving via a different neighbour of S, defines a set of node-disjoint paths from ***I*** to ***S***.

Proof: We prove by contradiction. Suppose that the paths taken by two copies of *m* received at ***I*** via different neighbors of ***S*** have a common node ***J***. This implies that ***J*** must have transmitted at least two copies of *m*, each received via a different neighbour of ***S***. But in the flooding algorithm, each node transmits the message at most once, a contradiction.

Unfortunately, this proof is flawed. Consider the network in Figure 8, where node 1 floods a packet in search of routes to node 5. Node 2 receives and rebroadcasts the packet. Note that node 2 only received and handled the packet once. Both nodes 3 and 4 receive the packet, and rebroadcast the packet to be received by node 5. Node 5 will now have 2 routes to node 1. These routes share a node (node 2) and a link (link 1→2), and are therefore non-disjoint.

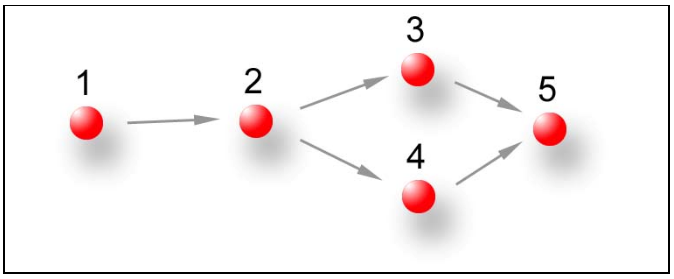


Figure 8: Non-disjoint routes

When a node in the AOMDV network receives multiple copies of a RREQ packet with the same sequence number, the node processes all received copies, since these copies may contain different routes to the source. However, this may create transmission loops. If node 1 floods a RREQ packet, it is received and rebroadcast by node 2, and node 1 receives the packet again, then node 1 should not create a route from this packet. AOMDV uses the packet's sequence number and hopcount to enforce loop freedom. A node in the AOMDV network only creates or updates a route if:

$$(-seqnum_i^d, hopcount_i^d) > (-seqnum_j^d, hopcount_j^d) \tag{1.1}$$

where:

$seqnum_i{}^d$	= the sequence number at node *i* for destination *d*.
$hopcount_i{}^d$	= the hopcount to destination *d* from node *i*.
node j	= the next hop from *i* to *d*.

The comparison is in the lexicographic sense. As a node is propagated through the network, its hopcount is incremented but its sequence number stays constant. If the packet visits the same node twice the node will already have a route with the same sequence number and a lower hopcount, and the update rule given above will prevent the node from adding the route again. A node receiving multiple RREQ packets from unique nodes which have valid routes (according to the update rule above) will respond to each RREQ with a RREP packet.

Marina and Das also introduce the notion of *advertised hopcount*. The advertised hopcount of node *i* for destination *d* is the maximum hopcount of the multiple paths to *d* available at *i*. The protocol only allows nodes to accept alternate routes to *d* with lower hopcounts.

Simulations performed by Marina and Das indicate that the AOMDV protocol produces around 20% less routing overhead than AODV while maintaining a higher byte delivery ratio than AODV. Additionally, the AOMDV protocol reduces end-to-end delay found in AODV considerably, often by more than a factor of two.

- **Advantages:** AOMDV offers improved byte delivery, routing overhead and end-to-end delay characteristics compared to AODV.
- **Disadvantages:** No clear disadvantages can be seen for the AOMDV protocol compared to other protocols in the reactive flat protocol class.

1.7.3 Dynamic Source Routing (DSR)

The key feature of Dynamic Source Routing (DSR) [55] is the use of source routing [56, 57]. Source routing is a routing technique in which the sender of a packet determines the complete sequence of nodes through which to forward the packet. The route is listed in the header of each packet that is transmitted, and intermediate nodes forward the packet by popping the next node address of the header in the packet. Two major drawbacks of source routing are the amount of bandwidth consumed by sending the entire route in each packet, and the amount of memory required in each node to keep a complete route to destination nodes. Source routing has been used in a number of contexts for routing in wired networks, using either statically defined or dynamically constructed source routes. In DSR, the source nodes determine routes dynamically and only as needed, there are no periodic broadcasts from routers.

DSR and AODV (AODV is described in section 1.7.1) share many features. When a node in a DSR network wishes to send a packet to a destination for which it does not have a route, it initiates a *route request* (RREQ) similar to AODV. While routes in AODV are set up in the reverse direction of the travel of the packet, routes in DSR are created by the packet simply adding each node that it hops through to its header. If the packet reaches the destination node, the route in its header will be the route to the destination. If it reaches an intermediate node with a route to the destination, the route in the RREQ packet's header will be appended with the route in the intermediate node's routing table (a variation of this method is used for networks with uni-directional links). DSR also uses the same route reply (RREP) packets as AODV, and the same route notification failure (RERR) mechanism as AODV.

Because the entire route is listed in a packet's header, no special mechanism is needed to eliminate transmission loops. Also, any node that forwards a packet caches the route contained in the packet for possible future use. Several very effective optimizations to source routing can be included as optional features in DSR, including:

- **Salvaging:** When a data packet meets a failed link on its route, an intermediate node can substitute a route from its own routing table in the packet's header.
- **Gratuitous route repair:** A source node receiving a RERR packet augments the next RREQ packet with the failed link information contained in the RERR packet (it "piggybacks" the RERR on the following RREQ). This ensures that other nodes are informed quicker of link failures, and helps to clean up the routing tables of other nodes in the network that may have the failed link in one of their source routes.
- **Promiscuous listening:** When a node receives a packet that is not addressed to itself ("overhears" the packet), it checks if it has a shorter route to the destination in its routing table. If not, it adds the route contained in the packet to its routing table. If it has a shorter route, the node sends a gratuitous RREP to the source node of the route with this new, better route. Promiscuous listening therefore helps to find shortest routes, and helps a node to learn different routes without directly participating in the routing process.

The main drawback of DSR is that, as a packet propagates through the network, the source route in its header can become huge. The routing information contained in the packet can become enough to exceed the accompanying message's usefulness.

- **Advantages:** DSR generates no routing overhead when there are no changes in the network. DSR can easily be adapted to support multiple routes to a destination.
- **Disadvantages:** DSR has large bandwidth overhead as a result of source routing, which means that DSR does not scale well to large networks. DSR also has a large memory overhead for nodes to record source routes to other nodes.

1.7.4 Temporally Ordered Routing Algorithm (TORA)

TORA [58, 59] is a loop-free, on-demand protocol that uses neither link state, distance vector or source routing principals, but a routing algorithm called *link reversal* (link reversal algorithms will not be discussed further in this work, the interested reader is referred to [60]). One of the key features of TORA is that the exchange of routing information is restricted to a region within one-hop distance of the node where the topological change occurred.

To create and maintain routes, each node creates a *directed acyclic graph* (DAG, see Figure 9) which starts at the node creating the graph. A directed acyclic graph is defined as "a directed graph with no path that starts and ends at the same vertex". A "height" metric is used to construct the graph, and is used to designate nodes as upstream or downstream, based on their relative heights on the graph. This is similar to the query/reply process found in LMR (LMR is not discussed in this work, the interested reader is referred to [61]).

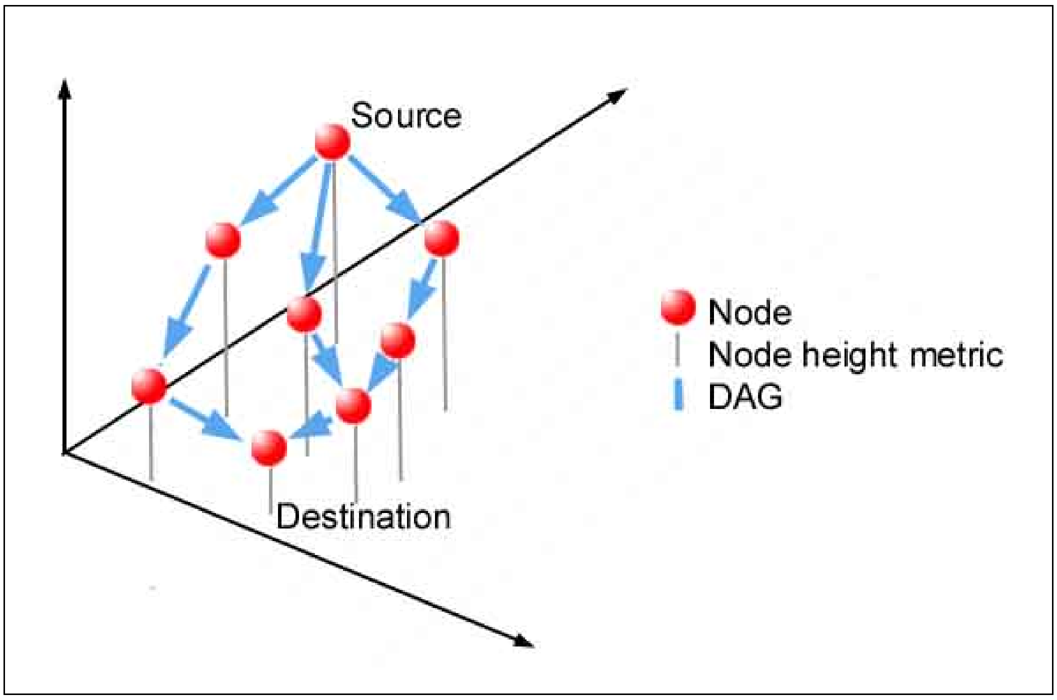


Figure 9: Height metric used to construct DAG in TORA

When a link is broken, the first upstream nodes from that node whose position and/or status have changed generates a new "height" metric for the node in question. The upstream nodes then propagate the new height level upwards along the DAG, ensuring a fast and structured update of routing information.

The flow of packets in TORA can best be described with an analogy to the flowing of water down a hill. The source node is at the top of the hill, and is connected through a network of tubes (wireless connections which form a DAG) to the destination node at the bottom of the hill. Water (packets of data) flow from the source node, through the tubes and intermediate nodes to the bottom destination node. The height of each intermediate node is calculated to ensure that the node is "lower" than its immediate upstream node. If a tube is blocked (a link is broken) the height of the node is set higher than any neighbouring node so that the water will flow back out of the node and find an alternative route to the destination (i.e. link reversal).

The height metric in TORA is dependent on the logical time of a link failure, which makes timing a crucial factor in TORA. TORA therefore assumes that all nodes are equipped with a synchronized clock (which is usually achieved by equipping each node with a Global Positioning System (GPS) device). More specifically, the height metric is computed using:

- The logical time of a link failure;
- The node ID of the node that defined the reference level;
- A reflection indicator bit;
- A propagation ordering parameter; and
- The node ID of the source node.

The first three factors above are used to calculate a reference level for a node. A new reference level is calculated each time that a downstream link from a node is broken. To erase paths that have been reported as broken, TORA floods the network with a *clear packet* (CLR). There is a potential for temporary transmission loops in TORA, especially when multiple groups of nodes are concurrently erasing routes and building new routes based on each other.

- **Advantages:** TORA provides multiple paths to each destination and has low routing overhead.
- **Disadvantages:** Temporary transmission loops can potentially occur.

1.7.5 Associativity-Based Routing (ABR)

ABR [62] uses none of the previously discussed routing principles. In stead, it defines a new metric called the *degree of association stability.* Association stability is defined as the degree of stability of a connection between two nodes over time. Each node periodically sends a one-hop beacon or hello message. All nodes receiving this message increments their association stability with this node. Over time, a high association stability metric should indicate a pair of slow (or synchronously) moving nodes. This association stability is used to select routes to a destination node. Associativity counts are reset when a beacon packet sent to a node fails. Establishing longer lived routes is a fundamental objective of ABR.

Route discovery in ABR is achieved by the node entering a *broadcast query and await reply* (BQ-REPLY) cycle. The node multicasts a BQ packet in search of nodes that have a route to the destination (or the destination itself). Any node that receives a BQ packet appends its address, associativity ticks with its neighbours, and QoS information to the packet, and then rebroadcast the packet. Any consecutive node receiving the packet will erase the information of the upstream node, retaining only the associativity count and QoS information between the upstream node and itself. When a packet reaches the destination, the packet contains a list of all the associativity information of the route that it took from the source to the destination. The destination node then selects the best route from itself back to the source node by choosing the route with the best associativity. If more than one route has the same associativity, the destination node choses the route with the shortest hop-count. Note here that the associativity count ("long-livedness" of the route) is more important than the length of the route. The destination node then uses the selected route to send a REPLY packet back to the source node. All the nodes receiving this REPLY packet on the route to the source marks their routes to the destination node as valid, and keeps other routes that they may have recorded invalid.

If a node moves and requires a new route, it will initiate a new BQ-REPLY cycle. If a node notices that an upstream node is not available anymore (via the beacon message) the node will erase any active routes containing that unavailable node and send a route erasure message to the downstream nodes on that route. If a node upstream from the destination notices that a downstream node is no longer available, the node erases the route it has to the downstream node and starts a *localized query* (LQ) process. If the destination node receives an LQ packet, the destination node sends a REPLY packet containing the best route between itself and the node that initiated the LQ. If the LQ fails (by timing out at the initiating node), the initiating node notifies the immediate upstream node of the failure. This upstream node repeats the same procedure. If this process continues unsuccessfully half the way back to the source node, the entire route will be erased and the source node can initiate a new BQ-REPLY cycle if needed. If any discovered route is no longer needed by the initiating node, the initiating node sends a *route delete* (RD) message. This message is not sent downstream along the route to be deleted, but is flooded to the network since the initiating node may not be aware of some downstream route changes.

An improvement to ABR, called Optimised Associativity-Based Routing (OABR) was introduced in [63]. One of the biggest improvements of OABR is that destination nodes do not select routes with the highest associativity count, but rather the shortest route with an associativity count above a certain threshold.

- **Advantages:** Since stable routes are preferred, route discovery and route failures should occur less frequently.
- **Disadvantages:** ABR requires periodic beaconing overhead to maintain associativity counts. Beaconing also requires nodes to permanently stay active - nodes cannot go into a sleep cycle to conserve power. ABR maintains only a single route to each destination.

1.7.6 Signal Stability based Adaptive Routing (SSA)

SSA [64] is a descendant of the ABR protocol (ABR is discussed in section 1.7.5), but replaces the associativity count used in ABR with the signal strength of the connection between nodes and the stability of a node's location to determine routes. SSA therefore also aims at establishing long-lived routes rather than shortest routes. SSA has two functional layers – a *dynamic routing protocol* (DRP) and a *static routing protocol* (SRP).

The dynamic routing protocol maintains the *routing table* (RT) and the *signal stability table* (SST). The signal stability information is obtained by a link layer functionality that measures the signal strength of periodic beacon messages, and the dynamic routing protocol classifies each link with a neighbour as a *strong channel* or *weak channel*. All incoming messages are received by the dynamic routing protocol, which passes the packet to the static routing protocol after updating the signal stability table. When the static routing protocol receives a packet, the protocol checks if the packet is destined for this node. If the packet is destined for this node, the protocol passes the packet to the node's packet stack. If the packet is not destined for this node, the protocol attempts to forward the packet to the intended destination. If the routing table does not contain any entries for the destination, a route search is initiated. When sending a route request packet, a node can specify whether a route consisting of arbitrary link strength is sufficient, or whether the route has to contain only strong channel links. Route request packets are multicast to all other nodes in the network. If a node receives a packet requesting a strong channel route over a link that is classified as a weak channel, the node will discard the packet. This ensures that only route request packets that satisfy the link requirement will ultimately reach the destination. The destination node responds to the first route request packet it receives, as the route followed by this packet is probably the shortest and/or least congested. If the source node does not receive a response to a request for a strong channel route within a specified time, it will send a request for a weak channel route (as this may be the only route available). If a broken link is detected during the transmission of a packet, the intermediate nodes will send a route error message upstream to the source node, indicating which link has failed. The source node will then a) Multicast a route erase message to notify all nodes that this route should be erased; and b) Initiate a new route search if needed.

Dube *et al.* [64] also suggest two improvements over the standard protocol, namely:

- To implement a third channel strength requirement which only *prefers* strong links, but does not require them. If this strategy is used, the strength of all channels traversed by a route request packet is recorded in the packet's header. A destination node receiving a route request packet does not respond to it immediately but waits for a certain period, and then chooses the route with the strongest channels.
- To allow intermediate nodes to respond to route request packets to destinations for which they have a valid route (called *gratuitous route reply*). In the standard SSA protocol intermediate nodes cannot attempt to repair routes – the route is deleted and the source node has to initiate a new route search. This may lead to longer delays in creating routes, though SSA compensates for this by explicitly aiming to create longer-lived routes. Allowing intermediate nodes to respond to route requests may lead to significantly reduced route request delays.

A drawback of the signal strength criteria is that it favors short hop distances. The effect of this is that packets take longer to reach their destinations (as a result of the increased hop-count), and that the transmission ranges of nodes are used inefficiently.

- **Advantages:** Since stable routes are preferred in SSA, route discovery and route failures should occur less frequently.
- **Disadvantages:** SSA has large delays during route requests and route failures, and node transmission ranges are used inefficiently. SSA also requires each node to be equipped with a hardware-level signal strength indicator.

1.8 Proactive/Reactive Hybrid Flat Protocols

Proactive/reactive hybrid flat protocols do not partition the network into clusters. This class of protocols tries to combine the best characteristics of proactive and reactive protocols by having nodes selectively use either proactive or reactive routing. Nodes usually keep routes to nearby nodes proactively, while only acquiring routes to far-away nodes when they are needed.

1.8.1 Zone Routing Protocol (ZRP)

In ZRP [65], each node proactively maintains routes to nodes within a certain zone. The zone of a node is defined as all the neighbouring nodes within a specified number of hops from the node, similar to the "fisheye" in FSR. This means that routes to nodes within this zone are immediately available. For destinations outside the node's zone, routes are determined reactively (on-demand), using any on-demand routing protocol. ZRP reduces routing overhead significantly compared to pure proactive protocols. Additionally, the delays associated with route discovery to destinations outside the node's zone are much less than in pure reactive protocols such as AODV or DSR. This is because the route request packet only has to travel to the *border* of the zone containing the destination node, and not to the node itself. Since each node in the destination node's zone maintains a route to other nodes in its zone proactively, the route to the destination node will be available immediately at any border node. However, the zone diameter have to be tuned for every network – if the zone diameter is too large ZRP will behave like a pure proactive protocol, if the zone diameter is too small ZRP will essentially be a pure reactive protocol.

- **Advantages:** ZRP is very efficient for networks where nodes communicate more frequently with neighbours than with faraway nodes, and network overhead can be traded off against delay times by tuning zone diameters.
- **Disadvantages:** Zone diameters in ZRP have to be tuned for each application, and the efficiency of ZRP is decreased by many overlapping zones.

1.9 Conclusion

This chapter provided an overview of the algorithms used to route data packets in mobile ad hoc networks. Some algorithms divide the network into hierarchical clusters, while others use a non-hierarchical or flat structure. This distinction was used as the main axis along which to classify ad hoc routing algorithms. Furthermore, some algorithms continuously maintain routes to all possible locations in the networks, while others only acquire the routes when they are needed. This distinction was used as the secondary axis along which to classify ad hoc routing algorithms. The chapter then provided overviews of algorithms that fall under each category, with advantages and disadvantages of each algorithm.

The next chapter introduces Swarm Intelligence and Ant Colony Optimisation. These concepts will be used to construct a new breed of highly adaptive ad hoc routing algorithms.

2 Ant Colony Optimisation

This chapter begins by introducing the concept of swarm intelligence along with the advantages and uses of swarm intelligence. The chapter then moves on to introduce the ant colony as a form of swarm intelligence, and elaborates on the history and mathematical representation of the ant colony optimisation metaheuristic. The chapter concludes by discussing why the ant colony optimisation metaheuristic is a suitable candidate for solving the routing problem in mobile ad hoc networks.

2.1 Introduction

Less than a hundred years ago the building of termite mounds, the nest-building of the social wasp and the ability of ants to converge on sources of food were considered a somewhat magical aspect of nature. How could these simple, seemingly uncommunicative creatures be responsible for such epic feats of construction and organization? Biologists have, over the last century, unraveled many of these mysteries and provided the foundation for fields of research variously known as Collective Intelligence, Swarm Intelligence and Emergent Behaviour.

The rest of this chapter is organized as follows. Section 2.1.1 introduces stigmergy and self-organisation, two fundamental enabling mechanisms behind swarm intelligence, after which swarm intelligence itself is introduced in section 2.1.2. The swarm intelligent mechanisms present in natural ant colonies are presented in section 2.2, while section 2.3 briefly looks at the history of a powerful class of optimising algorithms that was derived from the behaviour of natural ant colonies, namely Ant Colony Optimisation (ACO) algorithms. Section 2.4 introduces the mathematical representation and notation of the ACO algorithm employed by this work. Section 2.5 discusses why the ACO metaheuristic is a suitable mechanism for solving the routing problem in ad hoc networks, and section 2.6 concludes this chapter.

2.1.1 Stigmergy and Self-Organisation

One of the first publications on the study of biological swarms was made by the South African poet and naturalist Eugène Marais (1872-1936) who published “The Soul of the White Ant” [66] in 1925. In 1927, a Belgian author, Maurice Maeterlinck (1862-1949) published “The Life of the White Ant” [67] which was allegedly plagiarised from Marais' articles ([68] and discussion in [69]).

Although Marais had written a detailed document on the behaviour of termites (which were called white ants in the early 1900's), he was unaware of the mechanics of termite communication. The answer to this question was first documented by the French biologist Pierre-Paul Grassé. In 1959, Grassé published the results of a study of termites in which he noted that the termites tended to follow a set of very simple rules when constructing their nest [70]:

- First, the termites move around at random, dropping pellets of chewed earth and saliva on any slightly elevated patch of ground they encounter. This soon causes small heaps of moist earth to form.
- These heaps of salivated earth encourage other termites to drop more pellets in the same place. Soon, the biggest heaps start to develop into columns which will continue to be built until a certain height is reached.
- If a column was built close enough to other columns, the termites will start climbing each column and start building diagonally towards the neighbouring columns.

This revealed a key concept of the collective intelligence of insects: The actions of the termites are not governed from start to end by a purposeful plan, but instead is the result of how the termite reacts to its immediate environment, its "world". The termite does not need global knowledge, central communication or coordination. It just needs to follow a simple set of rules dependent on the state of its immediate environment. This process, called *stigmergy,* is defined by Grassé in the following way:

"The coordination of tasks and the regulation of constructions do not depend directly on the workers, but on the constructions themselves. The worker does not direct his work, but is guided by it. It is to this special form of stimulation that we give the name STIGMERGY (stigma: goad; ergon: work, product of labour = stimulating product of labour)."

The English summary of Grassé's work perhaps expresses the concept more directly: "The stimulation of the workers by the very performances they have achieved is a significant one inducing accurate and adaptable response, and has been named stigmergy." Stigmergy can be classified into the following two categories [71]:

- **Sematectonic or task-related stigmergy:** In *sematectonic stigmergy* the actions of the agent change the physical characteristics of the environment, such as adding a ball of mud or digging a hole, and the perception of the changed environment causes the next agent to perform the same (or related) actions, such as adding another ball of mud or enlarging the hole. This is similar to *sematectonic communication* in socio-biology [72].
- **Sign-based stigmergy:** In *sign-based stigmergy* the actions of the agent change something in the environment which does not make any direct contribution to the task but

is exclusively used to influence subsequent agents to perform task related actions. The depositing of pheromone by ants that lead to trail following by other ants is an excellent example of this [71].

Stigmergy, therefore, is a mechanism which enables an environment to structure itself through the activities of the agents in the environment. The current state of the environment and the current state of the agents in the environment determine the future state of the environment and the future states of the agents. According to Bonabeau *et al* [73] any structure that emerges from this kind of repeated interaction can be said to develop through a process of self-organisation (SO). Bonabeau *et al* describe self-organisation as "…a set of dynamical mechanisms whereby structures appear at the global level of a system from interactions among its lower-level components. The rules specifying the interactions among the system's constituent units are executed on the basis of purely local information, without reference to the global pattern, which is an emergent property of the system rather than a property imposed upon the system by an external ordering influence."

Stigmergy and self-organisation depend on the following four principles, which will be explained in terms of the termite nest:

- **Positive Feedback:** A *positive feedback* or *autocatalytic* process is a process that reinforces itself in a way that causes very rapid convergence. The termites initially drop pellets randomly. As soon as a concentration of pellets occurs, the higher concentration of pheromone emitted by this cluster of pellets provides *positive feedback* to other workers to drop pellets on the same spot. This *autocatalytic* snowball effect leads to the formation of a hill of pellets.
- **Negative Feedback:** Pheromone evaporates over time. If only a few pellets are dropped on one spot, the pheromone will evaporate causing even fewer termites to drop pellets on that spot. This *negative feedback*, in the form of pheromone decay, helps the larger piles of pellets to grow by preventing smaller piles from continuing to attract termites. In general, negative feedback is used to remove old or poor solutions from the memory of the system. It is important for the rate of pheromone decay to be tuned to the problem that is being solved. If the pheromone decays too slow, bad solutions will remain in the system. If the pheromone decays too fast, good solutions may be deleted before they can be exploited by the positive feedback mechanism.
- **Randomness or the amplification of fluctuations:** The termite piles could start anywhere – their location is determined purely by chance. A small fluctuation in the behaviour of one termite could have a large influence on future events. This *randomness* is used to direct current situations to fit the environment as they evolve, or to allow the formation of new solutions.
- **Multiple Interactions:** Each pellet that is dropped changes the state of the pheromone as

perceived by the termites. Depending on the level of pheromone, the termites will interact with the nest in different ways. There are therefore *multiple interactions* between the environment and the agents acting upon it.

2.1.2 Swarm Intelligence

In the middle 1940's Walter, Wiener and Shannon studied robots that were equipped with touch and light sensors, and very simple behavioural rules [74]. When these robots were placed together, they exhibited "complex social behaviour" in response to each other's movements [75]. Further studies into the coordination and interaction of multiple intelligent agents were done in the field of Distributed Artificial Intelligence (DAI) since the early 1970's [76]. The idea that complexity at group level may be achievable with very simple individual agents and with no need for central control was central to these research efforts. The concept of swarm or group intelligence comes from this idea. Bonabeau *et al* [14] provide the following definition of swarm intelligence:

"Swarm intelligence (SI) is the property of a system whereby the collective behaviours of (unsophisticated) agents interacting locally with their environment cause coherent functional global patterns to emerge."

Kassabalidis *et al* [77] describe the fundamental value of SI, which is that "SI provides a basis with which it is possible to explore collective (or distributed) problem solving without centralised control or the provision of a global model." Swarm intelligent behaviour in nature is responsible for phenomena such as the flocking of birds or schooling of fish [78], the formation of living bridges or chains by ants [79], collective hunting of hawks [80], the collection of dead ants in ant-nests into cemeteries [81] and the building of wasp nests [82]. It seems logical that if nature could find such unique and powerful applications for swarm intelligence, humans should be able to apply the same principles in a number of areas. Swarm intelligence have indeed been used with great success in a number of applications including optimisation problems (both in the form of Particle Swarm Optimisers (PSOs) [83] and Ant Colony Optimisation algorithms [84]; PSOs will not be discussed in this work), routing in communication networks [85], task allocation in multi-robot systems [74, 86], self-configuration of robots in factories [87], exploratory data analysis [88] and game learning [89].

It seems that the saying "Go to the ant, you sluggard; consider its ways and be wise!" (Christian Bible, Proverbs 6:6) is gaining a whole new significance. The next section introduces swarm intelligence as present specifically in the ant colony.

2.2 Swarm Intelligence and the Ant Colony

Ants have always been a fascinating subject for human beings. Several scientific books [90, 91] and pure literature books [66] on ants have met with surprising public success. Individually, they are remarkably simple creatures with limited memory and behaviour that sometimes seems to have a random component. But collectively, ants consistently achieve remarkable feats of cooperation, coordination and construction. Documented examples of these achievements include [92]:

- Emigration of a colony;
- Forming of chain-bridges;
- Building the ant nest;
- Preferentially exploiting the richest available food source;
- Finding shortest routes from food sources to the nest;
- Regulating the nest temperature within limits of one degree Celsius;
- Co-operating to carry or move large items; and
- Sorting items in the ant nest.

Of particular interest to this work is how ants can find shortest routes between their nest and food sources. To study this phenomenon, Deneubourg *et al* [93] set up an experiment that came to be known as the binary bridge experiment.

In the binary bridge experiment (see Figure 10), a food source and a nest of *Linepithema humile* ants are separated by a bridge with two branches (branch A and branch B), and with branch A longer than branch B.

In this species of ant, the ants deposit a substance called *pheromone* on the ground as they walk. Other ants can smell this pheromone, and their path-taking decisions are influenced by the pheromone. When subsequent ants sense this trail of pheromone, they will follow the trail with a probability that is proportional to the concentration of pheromone – the higher the concentration, the larger the probability that subsequent ants will follow this trail. The pheromone evaporates at an exponential rate, which means that the strength of pheromone encountered by another ant is a function of the original pheromone strength and the time since the trail was laid.

At the start of the experiment, ants randomly choose which one of the two branches to take. The ants that happened to take the shorter branch B will reach the food source first. As a result, these ants will return to the nest first, and enforce the trail of pheromone on their way back. The next ant leaving the nest is therefore more probable to choose branch B, since the concentration of pheromone on this branch is higher than on branch A. This autocatalytic process has been called the *differential length effect* [17], and will soon cause the ants to converge to branch B. Note that not *all* the ants will take branch B after convergence. The ant's decision function is stochastic, and the pheromone only produces a higher *probability* that the ant will take branch B.

A few points are worth noting regarding the binary bridge experiment:

- If the shorter branch B is only presented to the nest after some time, then the ants will not converge to branch B since the concentration of pheromone on branch A is already much higher than on branch B. This is called the *shortcut problem* [94].
- If the branch taken by most ants after convergence is suddenly blocked off, it can take a relatively long time for the ants to find a new route. This is called the *blocking problem* [94].
- If the two branches A and B are the same length, statistical fluctuations will cause one of the two branches to be chosen by a few more ants than the other. This will cause a slightly higher concentration of pheromone on the one branch, which in turn will cause a few more ants to take this branch. After some time the ants will converge on this branch, as if this branch were shorter than the other.

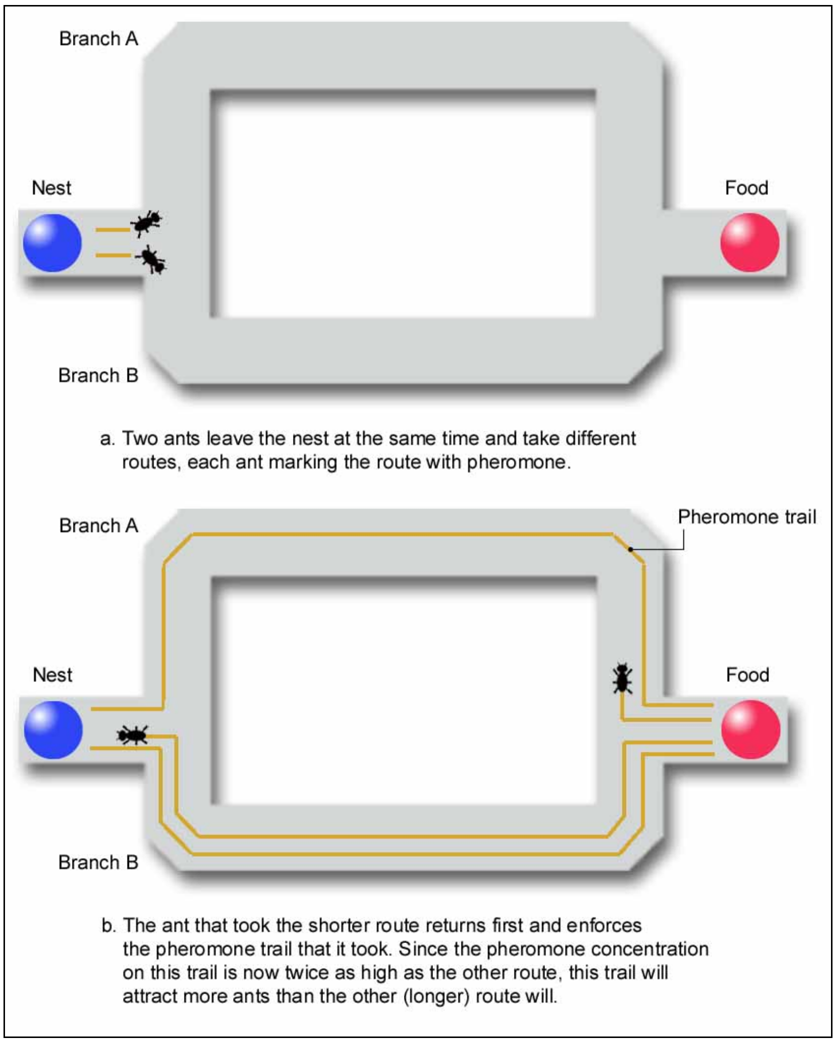

Branch A
Nest
Food
Branch B
a. Two ants leave the nest at the same time and take different routes, each ant marking the route with pheromone.
Branch A
Pheromone trail
Nest
Food
Branch B
b. The ant that took the shorter route returns first and enforces the pheromone trail that it took. Since the pheromone concentration on this trail is now twice as high as the other route, this trail will attract more ants than the other (longer) route will.

Figure 10: Binary bridge experiment setup

The differential length effect and pheromone-based autocatalysis form the base of a powerful suite of optimisation algorithms called Ant Colony Optimisation (ACO) algorithms.

ACO algorithms are derived from the route finding mechanism of ants as described above, and was first proposed by Marco Dorigo in 1992 as part of his Ph.D thesis [95]. Dr. Dorigo received the 1996 Italian prize for Artificial Intelligence and in 2003 the European Commission's Marie Curie Excellence Award for his work on ACO algorithms. The next section briefly looks at the history of ACO algorithms.

2.3 History of ACO algorithms

The earliest published accounts of ACO algorithms are two works by Maniezzo, Colorni and Dorigo [84, 96]. Their first paper introduces an optimisation approach based on the behaviour of ants, and describes three algorithms for solving the Traveling Salesman Problem (TSP). Maniezzo, Colorni and Dorigo called these algorithms ant-density, ant-quantity and ant-cycle [84]. Of these three algorithms, ant-cycle was the most successful, and the authors found optimal tour lengths for 30-city TSPs, as well as near-optimal tour lengths for 50 and 75-city TSPs. The same authors give a comprehensive overview of ACO as applied to the TSP in [97], which also compares the ant-cycle algorithm with tabu search and simulated annealing. Maniezzo, Colorni and Dorigo also describe the use of ACO algorithms for the asymmetric TSP, the quadratic assignment problem and job-shop scheduling problems.

The ant-system algorithm have been successfully applied to numerous combinatorial problems such as the Traveling Salesman Problem (TSP) [98], the Multiple Knapsack Problem (MKP) , the Bin Packing and the Cutting Stock problem [99], the Single Machine Total Weighted Tardiness (SMTWT) problem [100] and the Quadratic Assignment Problem (QAP) [101]. Table 1 presents a chronological overview of ACO algorithms and their applications (uses of ACO in ad hoc networks are omitted here as they will be discussed in Chapter 3; the table is adapted from [17]).

Table 1: Chronological overview of ACO algorithms and their applications

Year	***Authors***	***Problem name***	***Reference***
1992	Maniezzo, Colorni & Dorigo	Travelling Salesman	[96]
1994	Maniezzo, Colorni & Dorigo	Quadratic Assignment	[101]
	Maniezzo, Colorni & Dorigo	Scheduling Problems	[100]
1995	Gambardella & Dorigo	Travelling Salesman	[102]
1996	Gambardella & Dorigo	Travelling Salesman	[103]
1997	Gambardella, Taillard & Dorigo	Quadratic Assignment	[104]
	Bullnheimer, Hartl & Strauss	Vehicle Routing	[105]
	Gambardella & Dorigo	Sequential Ordering	[106]
	Costa & Hertz	Graph Colouring	[107]
1998	Maniezo	Quadratic Assignment	[108]
	Michel & Middendorf	Shortest Common Supersequence	[109]
	Maniezzo & Carbonaro	Frequency Assignment	[110]
	Lourenco & Serra	General Assignment	[111]
1999	Bauer *et al*	Scheduling Problems	[112]
	Gambardella, Taillard & Agazzi	Vehicle Routing	[113]
	Leguizamon & Michalewicz	Multiple Knapsack	[114]
	Liang & Smith	Redundancy Allocation	[115]
2000	Solnon	Constraint Satisfaction	[116]
2001	Cicirello & Smith	Distributed Factory Control	[117]
2002	Parpinelli Lopes & Freitas	Data Mining	[118]
	Parpinelli Lopes & Freitas	Classification Rule Discovery	[119]
	Broggi & Fascioli	Artificial Vision	[120]
	Labroche, Monmarche & Venturini	Data Clustering	[121]
2003	Ouiddir *et al*	Multi-State Power System Design	[122]
	Fenet & Solnon	Maximum Clique Problem	[123]
	Korosec *et al*	Mesh partitioning	[124]
2004	Guo *et al*	Most Probable Explanation Problem	[125]
	Jensen & Shen	Finding rough set reducts	[126]
	Nourelfath & Nahas	Redundancy Allocation for Multi-state Systems	[127]
	Oakes	Stylometry	[128]
	Green *et al*	Automatic programming	[129]
2005	Chen & Cheng	Scheduling Problem for Multiprocessors	[130]
	Karadimas *et al*	Routing identification in urban waste collection	[131]
	Christodoulou	Optimal Truss Design	[132]
	Jalali *et al*	Reservoir Operation	[133]
	Levanova	Simple Plant Location Problem	[134]
	Garlick & Barr	Dynamic wavelength routing	[135]

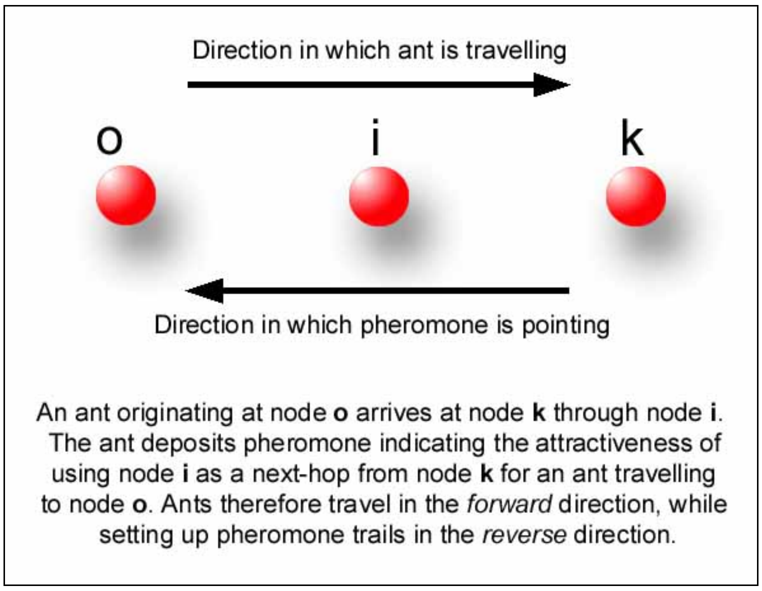
This section provided a brief overview of the historic development of ACOs. The next section introduces the general mathematical representation of ACO algorithms.

2.4 Problem Representation

This section introduces the general mathematical representation of ACO algorithms as used for solving the routing problem in ad hoc networks. Most ACO-based routing algorithms model each packet of information arriving at a node as an artificial ant. Ants are created by a node that generates traffic, and die when they reach a specified destination node. Each ant performs three distinct functions:

1. The ant deposits an amount τ of artificial pheromone on certain nodes,
2. at each node, the ant makes a stochastic decision on its next-hop destination based on a probability function p, and
3. the ant optionally distributes application and heuristic information through the network.

Consider a set of nodes $\boldsymbol{S}$ that form a network. At each node s, assume that a subset of nodes $\boldsymbol{N}_s$ that are known to be one-hop neighbours of node s is defined. Since the neighbours of each node will change as nodes move around, this set is a function of the time index t, i.e. $\boldsymbol{N}_s(t)$. Each node maintains a data structure containing a pheromone value τ_n^d for each destination/next-hop neighbour (d,n) pair.


Direction in which ant is travelling
o
i
k
Direction in which pheromone is pointing

An ant originating at node **o** arrives at node **k** through node **i**. The ant deposits pheromone indicating the attractiveness of using node **i** as a next-hop from node **k** for an ant travelling to node **o**. Ants therefore travel in the *forward* direction, while setting up pheromone trails in the *reverse* direction.

Figure 11: Pheromone depositing by ants

An ant originating at node o, arriving at node k through node i (see Figure 11) increases the amount of pheromone (or deposits pheromone) associated with the transition from node k to node i for an ant with destination node o. Ants can therefore be said to travel in the *forward* direction, while setting up pheromone trails in the *reverse* direction. This pheromone level at node k is indicated by:

$$\tau_{ki}^{o}$$

where:

o	=	the origin node, i.e. the node where this ant originated. This pheromone level will influence ants with node o as destination.
k	=	the current node.
i	=	the next-hop node with which this pheromone level is associated.

An ant arriving at node k at time t increases the pheromone by an amount $\Delta\tau$, so that:

$$\tau_{ki}^{o}(t+1) = \tau_{ki}^{o}(t) + \mathrm{V}\tau \qquad (2.1)$$

For the binary bridge experiment in section 2.2, Pasteels *et al* found that the probability of an ant to choose path A at time t+1 is given by [136]:

$$p_A(t+1) = \frac{(c+n_A(t))^{\alpha}}{(c+n_A(t))^{\alpha} + (c+n_B(t))^{\alpha}} = 1 - p_B(t+1) \qquad (2.2)$$

where:

c	=	the degree of attraction of an unexplored branch.
α	=	the bias to using pheromone deposits in the decision process.
$n_A(t)$, $n_B(t)$	=	the number of ants on paths A and B respectively at time t.

Larger values of c create a higher probability that ants will follow the path with the most pheromone. Larger values of α makes the system more sensitive to pheromone deposits and increases the probability that an ant will follow the path with more pheromone, even if the difference in pheromone levels on the two paths are very small. Pasteels *et al* found that equation (2.2) with $c \approx 20$ and $\alpha \approx 2$ best approximated the experimentally observed data.

Using this concept of pheromone following by ants, Dorigo and Di Caro [137] created the first ant algorithm called *Simple Ant Colony Optimization (SACO).* The SACO algorithm was created to solve the problem of finding the shortest path between two nodes on a graph. In the SACO algorithm, a number of ants are placed on the source node of the graph. At each iteration of the algorithm, each ant incrementally constructs a path to the destination node. At each node, the ant uses the following transition probability to select the next node:

$$p_{kn}^{d}(t)=\begin{cases}\dfrac{\tau_{kn}^{\alpha}(t)}{\sum_{x\in N_k(t)}\tau_{kx}^{\alpha}(t)} & \text{if } j\in N_k(t)\\ 0 & \text{if } j\notin N_k(t)\end{cases} \tag{2.3}$$

where:

t	=	time index
$p_{kn}{}^{d}(t)$	=	the probability that an ant at node k at time t traveling to node d will select node n, $1\geq p\geq 0$.
α	=	a positive constant that amplifies the influence of pheromone concentrations.
$\tau_{kn}(t)$	=	the pheromone level at node k at time t associated with using node n as a next-hop destination.
$N_k(t)$	=	the set feasible nodes connected to node k at time t.

After all ants have constructed a path from the source to the destination nodes, the ants retrace their paths to the source node deterministically. At each visited node, each ant deposits an amount $\Delta\tau$ of pheromone:

$$\Delta\tau_{kn}^{d}(t)\propto\frac{1}{L^{d}(t)} \tag{2.4}$$

The amount of pheromone deposited is proportional to the quality of the route found by the ant depositing pheromone. Since the objective of this algorithm is to find the shortest paths between nodes, the quality of the route is inversely proportional to the route length, $L^{d}(t)$.

The practice of depositing an amount of pheromone which is a function of the solution quality diverges from the observed behaviour of real ants, as Pasteels *et al* reported that ants deposit a constant amount of pheromone [136]. Ant algorithms therefore employ two distinct forms of solution evaluation:

- **Explicit evaluation:** In algorithms using *explicit evaluation*, ants deposit an amount of pheromone proportional to the quality of the solution.
- **Implicit evaluation:** In algorithms using *implicit evaluation*, all ants deposit the same amount of pheromone and the algorithm exploits the *differential path length* effect to find

solutions.

As ad hoc networks have dynamic topologies, it is necessary to incorporate a mechanism for ‘forgetting’ old routes. In ACO algorithms this is done by ‘evaporating’ pheromone exponentially over time. In the SACO algorithm, the pheromone level on each link is decreased after each iteration of the algorithm using the following equation:

$$\tau_{kn}(t+1) = (1-\rho)\left[\tau_{kn}(t) + \Delta\tau_{kn}(t)\right] \tag{2.5}$$

where:

ρ	=	the pheromone evaporation constant, $0 \le \rho \le 1$
$\Delta\tau_{kn}(t)$	=	the amount by which the pheromone was increased using equation (2.4)

For small values of ρ pheromone evaporates slowly, and as more pheromone accumulates on the links the algorithm is likely to converge much faster. For larger values of ρ pheromone evaporates quicker, the ants’ search becomes more random, and consequently the ants explore more. For ρ=1 all pheromone evaporates on every iteration, and the algorithm is reduced to a random search algorithm. The value ρ therefore controls the influence of search history on the algorithm by allowing the algorithm to “forget” previous solutions.

2.5 Suitability of the ACO approach

The following factors contribute towards making the ACO approach a suitable candidate for solving the routing problem in wireless ad hoc networks:

- The main computational expense of the ACO metaheuristic lies in the updating of pheromone levels and calculation of probabilities. These calculations are performed only at specific locations (i.e, only at nodes, no central computational mechanism is used), and using only locally-known information. This fits in well with the spatially distributed computational power and absence of global information that are characteristic of ad hoc networks.
- Since the ant’s decisions at each node are stochastic, some ants will deviate from the route with the most pheromone. This simple mechanism ensures that the search-space is continuously explored for new or better solutions. Mechanisms can also be incorporated to further encourage searching for new solutions. Schoonderwoerd *et al* [71] suggests explicitly introducing an additional noise factor *f* – at every time-step an ant has a probability *f* of choosing a purely random next-hop node, and a probability (1-*f*) of choosing a next-hop according to the pheromone tables in the nodes.
- The depositing of pheromone in an ACO-based system can be influenced by heuristic information, which in the case of ad hoc networks are transient network characteristics

(such as battery and congestion levels). If pheromone is deposited as a weighted sum of all the characteristics, then the ACO metaheuristic can be seen to inherently support multi-objective route evaluation. Moreover, the weighting of all the input values offers a single interface through which the relative influence of all the factors can easily be adjusted.

- As the characteristics of the network (such as battery and congestion levels) change, the amount of pheromone deposited by the ants and the decisions made according to the pheromone continuously adjusts. As one route becomes less attractive, the difference in pheromone levels between that route and another (previously less-attractive) route will become less, and the ants will gradually start to favour the other route (or routes). This is an extremely powerful feature which implies that ACO-based algorithms not only support multiple routes to destinations, but also a natural, built-in load-balancing mechanism between the different routes.

2.6 Conclusion

This chapter introduced swarm intelligence and the specific swarm intelligent paradigm that is used in this work, namely the ACO metaheuristic. The chapter started by looking at the collective feats accomplished by swarms of termites, and how scientists discovered that these accomplishments emerged as the result of a group of agents following a relatively simple set of rules. The discovery of *stigmergy*, a form of communication through the environment in the absence of a central communications infrastructure was discussed thereafter.

The focus then shifted to ways in which humans are harnessing this knowledge of natural phenomena to solve optimisation problems, both in the form of PSOs and ACO algorithms. Finally, the chapter laid the mathematical foundation of the ACO-based algorithm that will be used in this work to route packets in ad hoc networks, and discussed why such an algorithm is likely to be successful.

An overview of the different ways in which swarm intelligent algorithms have been employed to solve various variations on the theme of routing in ad hoc networks is presented in the next chapter.

3 Ant Colony Optimisation in Ad Hoc Networks

This chapter presents an overview of the uses of ACO in ad hoc networks. The chapter begins with a brief classification of previous works. The uses of ACO in flat (or non-hierarchical) ad hoc networks, as well as hierarchical ad hoc networks are presented. The focus then shifts away from ACO-based routing algorithms, and looks at various other uses of swarm intelligence and swarm intelligent frameworks in ad hoc networks.

3.1 Introduction

ACO algorithms have been employed to solve numerous problems in ad hoc networks. Dorigo *et al* first combined ACO with source routing to produce AntNet [138], Schoonderwoerd *et al* combined ACO with distance vector routing to produce Ant-Based Control [71], and various other authors produced ACO-based routing algorithms with the focus on different factors such as Quality-of-Service [12] and clustering of large networks [139]. Other authors generalised the ACO metaheuristic to produce generic, multi-objective swarm intelligence frameworks for solving various ad hoc routing and management issues [13, 140, 141]. ACO algorithms have also been employed in generating minimum-power broadcast trees [142], wavelength allocation for multiple channel access [143], and assigning cells to switches in mobile networks [144] (although this is not necessarily an ad hoc network problem).

Previous works on ACO with regards to routing in ad hoc networks can broadly be classified into the following categories:

- Flat Routing Protocols;
- Hierarchical Routing Protocols;
- General Swarm Intelligence Frameworks; and
- Various Other Works.

The rest of this chapter uses the classification above to present the works in each category. Section 3.2 presents ACO algorithms as used in flat (i.e. non-hierarchical) routing protocols, and section 3.3 presents ACO algorithms as used in hierarchical routing protocols. Section 3.4 looks at general swarm intelligence frameworks, which use swarm intelligence principles but may not specifically follow the ACO metaheuristic. Section 3.5 looks at various other uses of ACO in ad hoc networks, and section 3.6 concludes this chapter.

3.2 Flat Routing Protocols

In flat networks the nodes are not clustered into any form of hierarchical structure, all nodes are homogeneous in terms of network responsibility and packet handling. The first protocols to use ACO used basic source routing or distance vector algorithms where the traditional routing tables were replaced with pheromone tables, while more advanced protocols also incorporated ad hoc characteristics such as mobility and congestion into the pheromone mechanism.

3.2.1 Ant-Based Control (ABC)

The first applications of ACO algorithms were to static problems, that is, the characteristics of the problem stay the same while it is being solved. The first application of ACO to dynamic problems, that is, problems whose characteristics change while being solved, is by Schoonderwoerd *et al* [71]. It is concerned with routing in packet-switched networks (e.g., classical telephone networks). Schoonderwoerd *et al* propose an algorithm called ant-based control (ABC), which is the first system that replaces routing tables with pheromone tables.

In ABC, each node has a pheromone table with probabilities for each other node in the network (which implies that each node is aware of each other node). Ants are launched periodically by each node, each with a random destination node. Ants move from node to node, selecting the next node to move to according to the probabilities in the pheromone tables for their destination node. Arriving at a node, the ants update the values of that node's pheromone table entries corresponding to their source node. This is analogous to the ants depositing different "flavours" of pheromone, and only ants that are traveling to the node from which these ants was launched will respond to this specific "flavour" of pheromone. The ants therefore alter the table to increase the pheromone level pointing to their previous node, for an ant traveling to the ant's source node. When the ants reach their destination, they die. ABC therefore uses a form of distance vector routing.

Schoonderwoerd *et al* start by setting all pheromone levels to an initial value. The system is then allowed to run for a fixed period before nodes start generating data packets, in order for the nodes to establish routes to other nodes. They implement a relative pheromone updating scheme by making the amount of pheromone updated by each ant inversely proportional to the "age" of the ant – defined as the ant's hop count from its original source. The rational behind this is that ants with shorter paths will have more influence on the routing tables, and vice versa. They also implement a mechanism for relieving congestion by delaying ants en route to a congested node. This gives the congested node time to decongest, and also "ages" the ants so that the pheromone deposited by them will be less – thereby decreasing the probability of future visits to the congested node.

Although ABC is a very simple algorithm, it demonstrated that the ACO metaheuristic can be effectively used for routing packets in packet-switched networks. Schoonderwoerd *et al* compares the performance of ABC with a routing system based on mobile agents, and an approach which uses fixed, non-adaptive routing tables algorithmically optimised to yield the shortest paths. Unfortunately, the results of the comparison are given only in terms of the number of lost calls (a call is lost when an attempt is made to send a packet to a node that does not have spare transmission capacity left), and not in terms of other accepted metrics for wireless network performance such as power consumption, packet delay and overhead ratio. Nevertheless, in all their results the ant-based algorithm performs better (in terms of lost calls) than the other two algorithms.

3.2.2 AntNet

In [138], Di Caro and Dorigo describe the application of ACO to dynamic routing in packet-switched networks, using an algorithm that they call AntNet. This is the first application where ACO is used for routing with a source routing mechanism, i.e. the entire route of the packet is listed in the header of the packet.

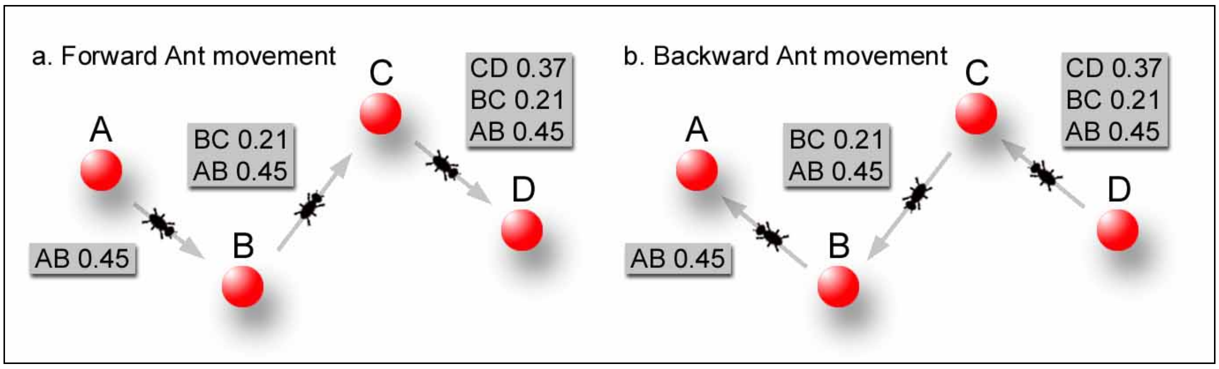


Figure 12: Forward and Backward ants in AntNet

Each node *s* in the AntNet network periodically generates an artificial ant (called a forward ant) with a randomly selected destination node *d* to observe the trip time from source node *s* to destination node *d*. The forward ant uses the current routing tables to find a path to node *d*, and records the route taken. When the forward ant arrives at the destination node, it creates another artificial ant (called a backward ant) which inherits the route listed in the header of the forward ant. After creating a backward ant, the forward ant dies (i.e. is deleted). The backward ant returns to node *s* to report the trip time, taking the same path as the forward ant in the opposite direction (see Figure 12).

Each node in the AntNet algorithm stores a routing table T_k which is organised as in distance-vector algorithms but storing probabilistic values. For each destination-neighbour (*d,n*) pair, T_k stores a probability value P_{nd} (see Table 2) which expresses the desirability of choosing *n* as a next-hop node to destination *d*, such that:

$$\sum_{n \in N_k} P_{nd} = 1, \quad d \in [1, N], \quad N_k = \{neighbors(k)\} \tag{3.1}$$

Table 2: Example of an AntNet routing table

Next-Hop Node:	Destination Node: A	B
E	0.43	0.57
F	0.11	0.88

Every time a forward ant whose destination node is *d* goes over a link from node *s* to node *n*, the corresponding routing probability is increased when a backward ant created by the forward ant returns to node *s* via node *n*. The amount of increase depends on the forward ant's trip time from node *s* to node *d*. Data packets are then forwarded to their destinations with the distance-vector routing tables created at intermediate nodes by the forward/backward ants.

The experimental results presented by Dorigo *et al* indicate that the AntNet algorithm provides better performance (in terms of average delay) compared to a suite of both Internet standard and state-of-the-art routing algorithms. Although AntNet forwards data packets using distance-vector principles, route discovery and maintenance are done using source routing. AntNet consequently suffers similar bandwidth and hop-count drawbacks encountered in Dynamic Source Routing (DSR). This translates to very limited scalability, and the results provided by Di Caro and Dorigo were consequently done on networks of only 36 nodes.

In [12], Oida and Sekido introduce an extended version of the AntNet system called Agent-based routing System (ARS). ARS keeps the basic source routing principles but adds hop-count and bandwidth restrictions to produce a Quality-of-Service (QoS) restricted routing algorithm. The following two restrictions were placed on forward ants in ARS:

- If a forward ant cannot select a next hop node since all outgoing links do not satisfy the bandwidth requirement, then the forward ant dies; and
- If a forward ant has visited the same node at least twice, then the forward ant dies.

Their simulations indicate that ARS achieves high *resource utilization* and low *resource contention*.

Another variation on AntNet is presented by Rajagopalan *et al* in [145], and is called Ad hoc Networking with Swarm Intelligence (ANSI). Rajagopalan *et al* call this work a *suite* of swarm-based routing algorithms, although it is actually one algorithm with user-specifiable underlying characteristics. In the ANSI network, each node has a proactive and a reactive component which can be switched on or off per node. Packets are forwarded using source-routing and pheromone tables similar to AntNet, and the AntNet FANT/BANT mechanism where packets record their routes in the packet headers are used for route discovery. The proactive/reactive logic is governed by the following rules:

- If node *S* has its proactive component turned on, then node S periodically broadcasts *proactive ants* to specific nodes in node *S*'s local area;
- When node *S* requires a route to destination node *D*, node *S* broadcasts a *forward reactive ant* to discover a route to node *D*;
- When node *D* receives a forward reactive ant from node *S*, it sends a *backward reactive ant* to node *S* using the source route from the forward reactive ant; and
- When a route fails at an intermediate node *I*, the ANSI protocol buffers the packets which could not be routed and initiates a route discovery to find destination node *D*. Node *I* also sends a *route error* message back to the source node *S*.

Rajagopalan *et al* test a version of ANSI that makes stochastic next-hop decisions and a version that makes deterministic next-hop decisions (i.e., always chooses the most attractive next-hop node) against the AODV routing algorithm. Their simulations indicate that AODV outperforms ANSI under most network conditions, except under conditions of very high mobility or very high network load. Although their simulations indicate that ANSI outperforms AODV under conditions of very high mobility or very high network load, the simulations were done on a network with a maximum of 4 hops between any two nodes. The largest known drawback of the source-routing mechanism used by ANSI is the amount of bandwidth consumed by listing the entire route in each packet. This effect will not be apparent in a network with only 4 hops between nodes, and where any route listed in the packet header can consequently only be 4 node addresses long.

Various variations of the AntNet algorithm have been introduced, such as the Probabilistic Emergent Routing Algorithm (PERA) for MANETs [146]. At the time of writing, AntNet is probably the best-known swarm intelligence routing algorithm and is very frequently used as a benchmark for other algorithms.

3.2.3 Ant-Colony Based Routing Algorithm (ARA)

Gues *et al* [147] presents an ACO-based routing scheme using distance vector routing. Route discovery in ARA is done by broadcasting Forward Ants (FANT), similar to Forward Ants in AntNet, and similar to Route Request (RREQ) packets in AODV. The FANT sets up a pheromone trail pointing back to the source node as it is broadcast through the network. When a route is found to the destination node, a Backward Ant (BANT), similar to Backward Ants in AntNet and Route Reply (RREP) packets in AODV is created. The BANT follows the pheromone trail created by the FANT back to the source node, and sets up a pheromone trail pointing to the destination node.

The amount of pheromone deposited by FANT and BANT packets is a function of the length of the route associated with the pheromone, and ARA also implements the packet sequencing mechanism (similar to DSDV and AODV) to prevent packet loops. However, in AODV each RREQ packet is also given a RREQ-sequence number to ensure that destinations only respond to a RREQ once. This mechanism is removed in ARA, and destinations will respond to multiple FANT packets received from the same node. In this way multiple routes are set up to destinations, and ARA therefore supports multipath routing (see Figure 13: Multipath routing in ARA).

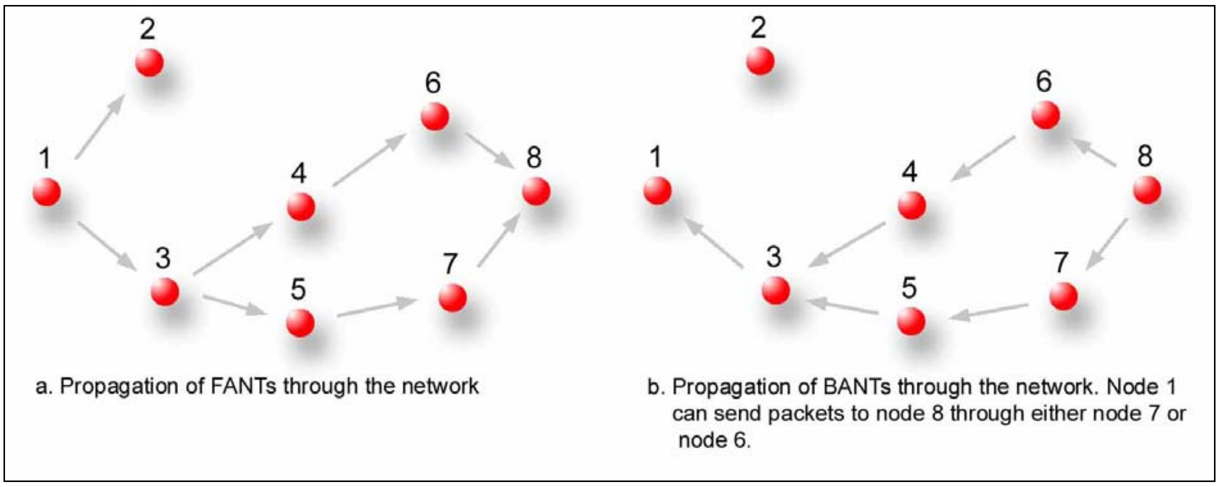

1
2
3
4
5
6
7
8
a. Propagation of FANTs through the network
1
2
3
4
5
6
7
8
b. Propagation of BANTs through the network. Node 1 can send packets to node 8 through either node 7 or node 6.

Figure 13: Multipath routing in ARA

After a route is established, the source node begins sending data packets. The data packets also reinforce the pheromone trails pointing to the source and destination nodes. In the case of a route failure, the intermediate node attempts to send the packet over an alternative link.

If no alternative link is found, the packet is returned to the previous up-stream node which also tries to send the packet over an alternative link. If the packet is returned in this way to the source node, a new route discovery sequence is initiated. This mechanism is generally referred to as *backtracking*.

The algorithm is compared to AODV, DSDR and DSR, and the results are given in terms of delivery rate (the ratio of packets a certain routing protocol was able to deliver successfully) and overhead ratio. Simulations performed by Gues *et al* indicate that:

- ARA and DSR perform comparatively in terms of *delivery rate*, with DSDR and AODV lagging behind; and
- ARA and AODV perform comparatively in terms of *overhead ratio*, with DSDR and DSR lagging behind.

3.2.4 Termite

Roth and Wicker [148] present an algorithm that is closely related to ARA, using the same distance vector routing principles. It differs slightly from ARA in terms of route discovery and failure recovery.

Route discovery in Termite is done by sending a Route Request (RREQ) packet. RREQ packets are not broadcast as in many other protocols, the source node explicitly generates the number of RREQ packets that it wants to send. Each RREQ packet chooses next-hop nodes using a uniformly distributed random decision function. This “random walk” of the packet continues until the packet is either deleted, or the packet finds a route (i.e. a pheromone trail) to the desired destination node. Note that since the packets are not broadcast but unicast to specific neighbouring nodes, this algorithm requires the use of hello messages to create the initial links between neighbour nodes.

A unique feature of this algorithm is the use of *seed* packets. Nodes can generate seed packets and send them on a random walk through the network, thereby advertising the existence of the node and setting up return pheromone paths to the source node. In a random walk, the packet randomly chooses a next-hop node from the nodes in the current node’s routing table, except for the link that it arrived on. This differs from the forward/backward ants in that no node will send a response to seed packets.

However, consider a node which has 3 neighbour nodes. Assume that one of the neighbour nodes transmitted a *hello* message so that the current node is now aware of this node. Since the current node is *not* aware of the other nodes, it cannot unicast a packet to them. A packet that is traversing the network on a random walk can only be transmitted to the one known node, and cannot discover the other two neighbour nodes. The nodes in a Termite network can therefore only discover neighbour nodes by using *hello* packets. However, the Termite algorithm only transmits *hello* packets when a node’s routing table is empty, and stops transmitting *hello* messages until its routing table is empty again. The random walk RREQ mechanism in Termite is thus not an effective mechanism for discovering routes.

If a RREQ packet finds a node with a route to the destination node, the node generates a Route Reply (RREP) packet, and sends the RREP packet back to the source node. The RREP packet follows the pheromone trail created by the RREQ packet back to the source node. The RREP packet also deposits pheromone on its way to the source node, and thereby sets up a route for the data packets to take.

RREP and data packets make probabilistic decisions at every node concerning their next-hop destination. The probability for each neighbour node is calculated using:

$$p_{nd} = \frac{\left(P_{nd} + K\right)^F}{\sum_{i \in N_k(t)} \left(P_{id} + K\right)^F} \tag{3.2}$$

where:

p_{nd}	=	the probability that a packet with destination node *d* will choose node *n* as a next-hop destination.
$N_k(t)$	=	the set known neighbours nodes of node *k* at time *t.*
P_{nd}	=	the amount of pheromone associated with a transition from the current node to node *n* for a packet with destination node *d.*
K	=	a constant that determines the sensitivity of the probability calculations to a small amount of pheromone, i.e. a pheromone-level *offset*.
F	=	a constant that accentuates the differences between pheromone levels on links. $F > 1$ accentuates the differences in pheromone between links, and $F < 1$ de-emphasises the difference.

When a packet deposits pheromone, the amount of pheromone is a pre-determined constant. Termite therefore does not support implementing different route metrics, or depositing pheromone proportional to the quality of the route. Pheromone levels are decreased every one second, though Roth and Wicker do not indicate why this interval was selected.

Roth and Wicker do not implement a sequencing mechanism (such as DSDV or AODV) to prevent packet loops, but make suggestions as to how this could be done. When an error occurs in the sending of a data packet the algorithm attempts one route discovery; if no route is found the packet is deleted. The algorithm therefore does not implement expanding ring searches, or a mechanism for sending the data packet to the first "upstream" node so that the upstream node may attempt route discovery.

Although Termite is a more advanced and scalable algorithm than AntNet, much work has to be done before it can be considered a mature algorithm. Roth and Wicker tested the algorithm in a simulated environment with 100 nodes, and give the algorithm's performance in terms of data goodput (the ratio of data bytes delivered to the overhead bytes required to deliver the data), mean path length, route confidence and node mobility. The algorithm is not compared to other ad hoc algorithms. Additionally, since the algorithm was simulated using a perfect MAC layer where any packet will be received if the sending node is within radio range of the receiving node, it still remains to be seen if the algorithm will be effective with a realistic MAC layer.

3.2.5 AntHocNet

Di Caro, Ducatelle and Gambardella [149] present a hybrid multipath algorithm that uses source routing principles combined with ACO.

Route discovery in AntHocNet is initiated reactively when node *s* requires routes to node *d*.

If node *s* does not have valid routing information to node *d*, node *s* sends out ant-like agents called *reactive forward ants* to discover routes to node *d*. If the current node visited by the forward ant has a route to node *d*, the packet is unicast using that route. If the current node does not have a route to node *d,* the packet is broadcast. Unlike the AODV algorithm which ignores any subsequent RREQ (or forward ant) packet received from the same generation, nodes in the AntHocNet network will compare both the travel time and hopcount of any subsequent forward ant, and will only rebroadcast the forward ant if both these criteria are within a certain factor of the best forward ant in that generation. What is not mentioned in the work is that this features may cause loops, and that it relies on the source route in the header of the forward ant to ensure routes are loop-free. A similar mechanism can therefore not be implemented in a network using distance-vector routing.

The forward ants gather route information as they travel towards node *d*, and at their arrival at node *d* become *backward ants* which trace their route back to node *s* and create routes toward the destination node *d* at the intermediate nodes they visit. Each node *i* stores a routing table T^i which contains, for each destination-neighbour pair known to node *i*, a measure T^i_{nd} of the desirability of using node *n* as a next-hop to destination *d*. The pheromone deposited by backward ants is calculated as the average of the inverse of the cost, in terms of estimated delay and number of hops, of traveling to node *d* through node *n*. Data packets select next-hop nodes stochastically with a probability P_{nd} proportional to the goodness T^i_{nd} of the link, calculated using:

$$P_{nd} = \frac{\tau^2_{nd}}{\sum_{i \in N_d} \tau^2_{id}} \tag{3.3}$$

Since the amount of pheromone by backward ants takes the packet delay into account, the AntHocNet algorithm features automatic *load balancing*. On routes with higher network load, packets will be delayed longer causing less pheromone to be deposited on these links. As the desirability of these links decrease, they will be used less and their congestion decreased.

Once routes are created and a data session is running, node *s* sends one *proactive forward ant* toward node *d* on every n^{th} data packet. These proactive forward ants use the same stochastic next-hop decision function as the data packets and can therefore monitor the quality of the routes in use. Additionally, at each node a proactive forward ant has a small probability of being broadcast so that it can explore new routes. To limit the proliferation of these forward ants through the network, the number of allowed broadcasts of each proactive forward ant is limited to two. An important consequence of this is that proactive forward ants can only discover new routes in the vicinity (within two hops) of existing routes, and cannot discover new routes outside this boundary. Di Caro *et al* also state that the purpose of proactive forward ants is to search for path improvements and variations, not to search for entirely new routes. The AntHocNet algorithm also uses *hello* messages to improve local connectivity of nodes, at the cost of slightly increased overhead. Di Caro *et al* draws a parallel between the local connectivity using *hello* messages, and pheromone diffusion in ant colonies where pheromone on the ground diffuses and can be detected by ants further away.

When a node becomes aware of a link failure on a route which has alternative next-hop nodes to the destination or which is not used regularly, the node updates its routing table and sends a notification, similar to the RERR packets used by AODV, to its neighbours. If the failure occurs on a route with does not have alternatives or which is used regularly, the node attempts to locally repair the route by broadcasting a *route repair ant* that travels to the destination in a similar way to a reactive forward ant. To limit the proliferation of route repair ants, the maximum number of allowed broadcasts of the packet is set to two. If no reply is received within a specified time, the entry is removed from the routing table and a notification is broadcast to all neighbour nodes. Di Caro *et al* does not mention whether a data packet on the failed route will be dropped at the node where the link failed or backtracked to an upstream node.

Di Caro *et al* tested the protocol in an environment with a realistic MAC layer and compared it to the AODV algorithm, and reported results in terms of *delivery ratio* and *end-to-end packet delay*. In all reported experiments, AntHocNet produces superior delivery ratio over AODV. In simpler scenarios (with less node mobility or fewer nodes) AODV produces lower packet delay than AntHocNet, but AntHocNet produces better packet delay in more complex scenarios.

Di Caro *et al* note that the superior delivery ratio and end-to-end delay of AntHocNet comes at the cost of higher routing overhead ratio – between 1.5 to 3 times higher than AODV. However, the control overhead is measured in packets. The control packets used by AODV have a static size, while the control packets used by AntHocNet grows due to source routing. Consider the case where node *s* requires a route to node *d*. Assuming that AODV RREQ and RREP packets are 24 and 20 bytes long respectively, that an IP address is 4 bytes long, and that the route from node *s* to node *d* contains 10 hops, the number of overhead bytes produced by one RREQ/RREP packet traveling from source to destination and back can be calculated as follows:

$$OverheadBytes(AODV) = (24 \cdot 10) + (20 \cdot 10) = 440 \text{ bytes} \tag{3.4}$$

Assuming that an AntHocNet reactive forward ant is originally 8 bytes long and grows with 6 bytes at each hop (4 bytes for an IP address, 2 bytes for route quality information), the number of overhead bytes produced by one forward and backward ant in AntHocNet can be calculated as follows:

$$OverheadBytes(AntHocNet) = 2\left(\sum_{n=1}^{n=10}(8+6n)\right) = 820 \text{ bytes} \tag{3.5}$$

The actual overhead in bytes resulting from the AntHocNet algorithm is roughly double the overhead in bytes created by the AODV algorithm, although it is only a single packet traveling from source to destination and back. The real routing overhead of AntHocNet measured in *control bytes* can therefore be expected to be roughly 3 to 6 times larger than that of AODV.

3.2.6 Ant-AODV

Shivanajay, Tham and Srinivasan [150] propose a proactive-reactive hybrid protocol which combine elements from AntNet and the AODV routing protocol. The Ant-AODV routing protocol maintains a population of forward ants which explore the network with a source-routing list of visited nodes in the packet's header. Additionally, when a node requires a route to a specific node it may reactively launch RREQ packets, though it is not clear whether the launched RREQ packets use source or distance-vector routing. The protocol also uses frequent hello messages which allow nodes to be continuously aware of their neighbours, and alerts nodes of link failures. Nodes maintain distance-vector routing tables with destination/next-hop pairs and associated hop count and sequence numbers for each route.

The protocol is compared to the AODV protocol and results reported in terms of end-to-end delay, packet delivery fraction, normalised routing overhead and node connectivity. Simulations performed by Shivanajay, Tham and Srinivasan indicate that the end-to-end delay and packet delivery fraction of Ant-AODV is comparable to AODV, with Ant-AODV having a slightly higher normalised routing overhead due to the continuous proactive movement of forward ants.

3.3 Hierarchical Routing Protocols

The flat routing protocols discussed in the previous section usually offer a relatively fast (in terms of packet delay) and computationally inexpensive routing protocol, but at the expense of scalability. Hierarchical networks improve the scalability of the network by dividing the nodes into clusters or into a hierarchy of clusters. Hierarchical networks are generally less suited for highly mobile environments, as the continuous updating of cluster memberships cause considerable computational and network bandwidth overhead. This section presents hierarchical routing algorithms that use swarm intelligence to discover and maintain routes.

3.3.1 Mobile Ants Based Routing (MABR)

Heissenbüttel and Braun [139] introduce MABR as the first routing algorithm for large-scale MANETs inspired by social insects. MABR is a hierarchical routing algorithm based on AntNet. Each MABR node is assumed to be aware of its geographical location by means of a location device such as GPS. Nodes are also assumed to be able to determine the position of all other nodes relatively accurately.

The algorithm consists of three components (alternatively called 'layers' or 'protocols' by different sources):

- **Topology Abstracting Protocol (TAP):** This layer groups the network into logical zones, using the positional information of the node to assign zone membership. Each node groups the network into logical zones with itself as the center node. MABR is therefore, strictly, not a hierarchical protocol, and there are no central nodes responsible for grouping the network into zones. Nodes close to the current node are grouped into small zones (i.e. less nodes per zone), and more nodes grouped together as the distance from the current node increases. Although used in a different context, this is reminiscent of the "fisheye" in FSR (see section 1.6.1).
- **Mobile Ants-Based Routing (MABR):** The MABR layer routes packets over the simplified hierarchical network. Each node stores two data structures – a *routing table,* and a *link-cost table.* The routing table contains a row for each outgoing link (i.e., each neighbour zone) and a column for each destination zone. The table stores the probability that a specific link will be used as a logical next-hop for a packet going to a specific logical zone. As packets are transmitted through the network, the route taken by the packet is recorded in the header of the packet (that is, a source routing mechanism is used). However, in order to limit the excessive packet sizes associated with source routing, the followed path is approximated by using a sequence of straight lines. To discover new routes, the same FANT/BANT mechanism as in AntNet is used, with BANTs updating pheromone levels based on the length of the route discovered by the FANT. When a node sends a packet, the node determines in which logical zone the destination is located, and then uses the probabilities in the routing table to select a logical next-hop zone. Since it is possible to have more than one route with a non-zero probability, MABR supports multi-path routing. Although Heissenbüttel and Braun do not formally introduce a load-balancing mechanism, they note that the link costs associated with next-hop probabilities could be used to implement such a mechanism.
- **Straight Packet Forwarding (SPF):** The SPF layer is responsible for the actual forwarding of packets from one node to the next. This algorithm can either select any node which is closer to the destination than the current node as next-hop, or the algorithm can select the node which reduces the distance between the current node and the destination the most.

The MABR algorithm is still a work in progress, and no full implementation of this protocol or comparisons with other protocols are available at the time of writing.

3.3.2 Adaptive-SDR

Kassaballidis *et al* [151] present a variation on the schemes used in AntNet and ABC, but tailored specifically for large-scale satellite networks. The Adaptive-SDR algorithm consists of three parts – clustering the network into colonies, finding network routes using ants, and forwarding packets using the routes discovered by the ants. The following paragraphs describe these three parts of Adaptive-SDR:

- **Clustering into Colonies:** Kassaballidis *et al* note that the traffic generated by AntNet and ABC when discovering routes using the FANT/BANT mechanism would be prohibitive in large-scale networks. Moreover, the large traveling times encountered by FANTs in large networks increases the likelihood that the information carried by BANTs are outdated. The network is therefore clustered into colonies. The number of colonies is chosen to minimize the number of ants required, and is calculated using:

$$NrOfAnts = NN\left(\frac{NN}{NC} - 1 + NC - 1\right) \tag{3.6}$$

where:

NN	= the number of nodes in the network.
NC	= the number of colonies in the network.

Minimising equation (3.6) yields the smallest number of ants where $NC = \sqrt{NN}$. K-means clustering with the Euclidian distance metric is then used to cluster the network. The clustering is done by a central clustering entity that is aware of the geographical locations of all the nodes. The clustering is not performed very frequently, only in the beginning stage of the algorithm and whenever the network topology changes enough to justify a re-clustering of the nodes.

- **Discovering Routes:** The route discovery mechanism uses two types of agents, called *colony ants* and *local ants.* Colony ants are responsible for finding routes from one cluster to another, while local ants are responsible for finding routes within a cluster. Each node maintains two routing tables, a *colony routing table* and a *local routing table* to facilitate routing between colonies and between nodes in a colony respectively. Nodes periodically send FANTs to each node in the node's local colony, and to one node in each outside colony. As FANTs are transmitted through the network, the route taken by the packet is recorded in the header of the packet (that is, a source routing mechanism is used). Once the destination node is used, a BANT is launched which traces the route taken by the FANT back to the source node, and updates pheromone levels on the way.
- **Packet Forwarding:** Data packets are forwarded using the probabilities created by forward and backward ants, using the colony routing table and local routing table for nodes outside and inside the current node's colony respectively. The protocol also monitors the length of the queue of packets destined for each next-hop node. The length of the packet queue is considered as an indication of the current level of congestion at the next-hop node, and the next-hop probabilities are adjusted to favour nodes with lower congestion. However, the algorithm does not provide facilities for using network

conditions other than the packet queue to influence next-hop decisions.

The algorithm is compared with the AntNet, Link-State (LS) and Distance-Vector (DV) algorithms on networks with 16 and 49 nodes. Their simulations indicate that Adaptive-SDR has higher delay times than the other algorithms, but provides higher data throughput and lower packet loss than the other algorithms.

3.4 General Swarm Intelligence Frameworks

The previous sections introduced specific swarm-intelligence based routing algorithms. This section discusses general frameworks that have been introduced to solve various network management and routing problems, and where a user-defined set of metrics and decision-functions can be used to make network and routing decisions.

3.4.1 Multi-Swarm Framework

In [140], White and Pagurek propose a general framework for solving various network routing and management issues using agents. In this system, ant-like agents solve various problems by moving through the network and interacting with chemical messages at nodes. Each chemical message consists of a label and a concentration. Each agent consists of:

- **Emitter(s):** Each agent is capable of ‘emitting’ certain chemicals. This is the first work where the capability of ants to emit pheromone is abstracted and extended to the capability of agents to emit various chemicals. For each chemical that the agent is capable of emitting, the agent is equipped with an Emitter Decision Function (EDF) that uses local information at the node to manage the quantity of the chemical emitted (i.e. deposited) by the agent.
- **Receptor(s):** Each agent is capable of sensing certain chemicals using *receptor(s)*, analogous to ants being able to smell pheromone. Each receptor have an associated Receptor Decision Function (RDF) that is used to determine the receptor’s sensitivity to the chemical, and it is possible to associate actions with the receptor.
- **Chemistry:** The evaporation of pheromone in natural ant systems is replaced by a generalised *chemistry* mechanism. This allows chemicals to react to other chemicals, to react to node conditions, or simply to evaporate as in the case of pheromone.
- **A Migration Decision Function (MDF):** The Migration Decision Function is a set of rules or a function that is used to decide which area or node the agent should visit next. This may be a hard-coded route, or a decision based on chemical concentrations and other node conditions.
- **Memory:** The agent’s *memory* stores the internal state of the agent and optionally concentrations of chemicals held internally by the agent.

The agents interact along swarm-intelligence principles to perform various tasks. The framework allows for multiple swarms using combinations of chemicals to communicate complex issues to other agents. Although limited implementation data are available, this work provides a valuable contribution in abstracting all the components of a swarm intelligent system and presenting them in a homogeneous manner.

In [141], White *et al* presents an implementation of the framework discussed above in an ad hoc network. The implementation uses various types of ant-like agents, namely:

- **Explorer** agents search for a path from a source to a destination;
- **Allocator** agents allocate resources on the links used in a path; and
- **Deallocator** agents de-allocate resources on the links used in a path.

White *et al* examine the viability of using these agents for multi-path and priority-based routing in a session-based environment (i.e. cell phone networks). The algorithm uses a form of source-routing to find viable paths, and agents evolve using genetic algorithms. White *et al* do not give a complete routing protocol; their efforts are focused on establishing a general framework to be used in the implementation of such a protocol.

3.5 Various Other Works

In [152], Kassabalidis *et al* propose modifications to ACO-based routing algorithms to minimise the power consumption of transmissions. The energy-cost of transmissions is calculated at each node, and the pheromone tables are adjusted to favour routes consuming less energy. As noted in [25], it is important that the energy-cost be calculated for the entire route and not on a hop-by-hop basis, as this will bias the system toward choosing shorter hops and not effectively using the transmission range of nodes. Kassabalidis *et al* therefore propose a scheme similar to AntNet where ants only update pheromone values after the complete trip to the destination node was made. However, Kassabalidis *et al* do not give details as to how the power consumption should be calculated at each node, or how the information should be incorporated into the pheromone tables.

Das *et al* [142] use ACO algorithms to construct minimum-power routing trees in wireless networks - the aim being to develop an algorithm for broadcasting a single packet of information to all nodes in the network using as little power as possible.

Although Cardoso *et al* [153] do not concern themselves with ad hoc networks, they present an implementation of the ACO algorithm to solve a multi-objective network routing problem. Cardoso *et al* are concerned with routing in traditional fixed networks, and introduce a *cost vector* containing a weighted sum of network inputs. A central planning entity uses this cost vector and global network knowledge to let the ACO algorithm incrementally build routes. Once satisfactory routes are found, the central planning entity can assign these routes by downloading the hard-coded routes into a node's memory.

In , Awerbuch *et al* [154] present an interesting variation on swarm intelligent routing that is resilient to a high level of proactive adversarial attack.

3.6 Conclusion

This chapter provided an overview of the uses of swarm intelligence in ad hoc networks. The first algorithms simply replaced the routing tables from conventional wired routing algorithms with pheromone tables. ABS used distance vector routing to route packets, with the pheromone deposited being a function of the route length. AntNet used source routing to route packets, and provides very reliable packet delivery but at the cost of increased network overhead. Various other routing algorithms were discussed that acquires routes either proactively, reactively, or in a predetermined hybrid combination.

However, most early routing algorithms are direct descendants of traditional wired network routing algorithms. The criteria for optimal routes in ad hoc networks are different than the criteria for optimal routes in traditional wired networks. Ad hoc network devices are usually portable, and depend on battery power to function. Various works were therefore aimed at using the ACO algorithm to minimise the power required to route packets, or to alter the routes taken by packets so that nodes with low battery levels are avoided. This was usually done by incorporating node or route battery levels into the ACO heuristics term, or by directly altering the routing probabilities calculated from pheromone levels so that routes with high battery levels are favoured.

Researchers also started looking at various other uses of swarm intelligence, and in particular ACO, in ad hoc networks. ACO have been employed to minimise power consumption, cluster networks into hierarchies, and even to protect the network from outside attacks. A general framework was discussed that abstracts the components of a swarm intelligent system to produce a system where agents deposit combinations of chemicals (i.e. pheromones), and perform various routing and network management actions in reaction to chemical levels.

The next chapter introduces metrics used to measure the performance of ad hoc networks.

4 Ad Hoc Network Performance Metrics

The word 'yardstick' is defined as "a test or standard used in measurement, comparison, or judgment" [155]. This short chapter introduces such yardsticks or metrics for measuring and comparing the performance of ad hoc networks. The chapter then introduces common features of ad hoc networks used to judge the suitability of ad hoc routing protocols, and concludes by selecting the metrics and features used to evaluate and characterise the performance of the ACODV routing algorithm.

4.1 Introduction

As far back in time as Noah's ark, the lack of a yardstick was arguably not a serious drawback. Most measuring was done by one craftsman completing one job at a time, and consequently the accuracy or even the length of measuring sticks did not make much difference.

The cubit of Noah's time was commonly defined as the distance from the tip of a man's elbow to the end of his middle finger. This definition was useful because the yardstick was (more or less) universally available and couldn't be mislaid. However, it was not a universally fixed dimension or a standard.

In 1672, Sir Isaac Newton noticed that when two flat pieces of glass were pressed together, circular bands of rainbow-like colors could be seen. Later, other scientists built on Newton's early findings to establish a new branch of science called *interferometry* [156]. This method of using a ray of light as a measuring stick enabled modern scientists to measure distances very accurately. However, accurately representing the measurements made by scientists requires an accurate *metric* of length that is known to other scientists. The "Systeme International d'Unites" (or SI system) is an international set of metrics (see Table 3) used to accurately express physical quantities.

To measure the performance of ad hoc networks, similar metrics have to be defined. Moreover, since the performance of any ad hoc network depends on the environment wherein the network operates, it is also necessary to define metrics that characterize the network environment. This work therefore defines the following two classes of ad hoc metrics:

- *Scenario Metrics* describe the environment in which the ad hoc network operates. Scenario metrics are therefore not measures of the protocol's performance, but are necessary parameters to compare the performance of protocols in different environments.

- *Performance Metrics* describe the actual performance of the protocol, given a set of scenario metrics.

Additionally, a set of qualitative and quantitative *Protocol Features* are defined that describe application independent features of a protocol, such as the maximum number of nodes supported by the protocol, and whether the protocol is regarded as pro-active or reactive. Note that although these features are independent of the application, they are dependent on the *scenario*, as described by the application's scenario metrics.

The rest of this chapter is organised as follows. Section 4.2 describes ad hoc scenario metrics, followed by performance metrics in section 4.3. Section 4.4 describes various qualitative protocol features. Finally, section 4.6 concludes this chapter.

Table 3: Definitions of SI base units

Unit	***Definition***
Meter	A meter is the length of the path traveled by light in a vacuum during a time interval of 1/299 792 458 of a second.
Kilogram	A kilogram is the mass of the International Prototype Kilogram, a cylinder of platinum-iridium alloy, stored at Seures, France, by the International Bureau of Weights and Measures. Since its installation in 1889 it has only been brought out 3 times to be cleaned and weighed. Eighty copies exist, of which 6 are deemed official. The last time the cylinders were removed and cleaned (between 1988 and 1992) a variation of 23 microgram was found, due to microscopic surface contamination and abrasion. The kilogram is unique among the seven base SI units in that it is the only unit still defined in terms of a physical artifact.
Second	A second is the duration of 9 192 631 770 periods of the radiation corresponding to the transition between the two hyperfine levels of the ground state of the cesium-133 atom.
Ampere	An ampere is that constant current which, if maintained in two straight parallel conductors of infinite length, of negligible circular cross section, and placed one meter apart in a vacuum, would produce between these conductors a force equal to 200 Newton per meter of length.
Kelvin	A kelvin is the fraction 1/273.16 of the thermodynamic temperature of the triple point of water.
Mole	A mole is the amount of substance of a system that contains as many elementary entities as there are atoms in 0.012 kg of carbon-12.
Candela	A candela is the luminous intensity, in a given direction, of a source that emits monochromatic radiation of frequency 540 THz and that has a radiant intensity in that direction of 1/683 watts per steradian.
Radian	A radian is the plane angle between two radii of a circle that cut off on the circumference an arc equal in length to the radius.

4.2 Scenario Metrics

Scenario metrics define the environment in which an ad hoc network functions. These metrics do not contribute to the performance evaluation of a network, but it is critical to consider these metrics to ensure comparable results for use in any performance evaluation or comparison.

4.2.1 Number of Nodes

The number of nodes that are present in the network. Note that not all the nodes present in the network are necessarily participating in the network's activities at all times, since some nodes may be in some form of a sleep mode.

4.2.2 Node Mobility

A mobility metric allows the effect of mobility on nodes to be quantified. A mobility metric should be protocol independent, i.e. the metric's implementation should not introduce a bias toward any routing protocol. Additionally, a mobility metric should be obtainable by real-world network nodes, as opposed to being purely an artifact of simulations. Such a metric is presented by Boleng *et al* in [157] using *link duration* (defined as the average time for which links are valid), which can be independently calculated by each node, as a mobility metric. Note that using link duration as an indication of node mobility also lets the metric indicate the network's rate of topological change.

4.2.3 Node Pause Time

The information supplied by a node mobility metric can be significantly enhanced by adding a *node pause time* metric. Node pause time is defined as the average time that a node in the network is expected to be stationary before starting to move again. The reason for this is that nodes that continuously move will affect a protocol different than nodes that move rapidly for a short time and then stay stationary for a prolonged period. This difference will not be discernable using only a node mobility metric. Node pause time can be defined in one of the following ways:

- The average fraction of time that any node is stationary, calculated as the total stationary time divided by the total mobile time of all nodes over a given period; or
- The average time that any node is stationary during a period of one hour.

This work uses the first definition. Note that this metric, calculated above for the entire network, can also be defined per node.

4.2.4 Degree of Connectivity

A node's *degree of connectivity* is defined as the average number of one-hop neighbours that a node has during a given period of time.

4.2.5 Physical Network Size

A network's *physical size* is defined as the physical area covered by a network. Although some applications (like satellite networks) require a three dimensional specification of the network size, the networks studied in this work only require a two dimensional network size specification. Network size is therefore specified in this work by specifying only the *length* and the *width* of a network.

4.2.6 Node Receive Distance

A node's *receive distance* is defined as the maximum distance that two nodes can be apart while reliably exchanging packets.

4.2.7 Link Speed

A node's *link speed* is defined as the effective transmission speed of the node's radio interface, measured in bits per second.

4.2.8 Packet Size

For the purposes of this work, the packet size in a network is defined as the average size of data packets that are sent through the network, measured in bytes.

4.2.9 Fraction of unidirectional links

The *fraction of unidirectional links* is defined as the overall fraction of unidirectional links in a network during a given period of time. This work assumes that links are always bidirectional, and therefore assumes that this fraction is always zero.

4.2.10 Fraction and frequency of sleeping nodes

Although the transmission of a packet of data consumes more power than receiving the same packet, only a very short time is spent by the node transmitting data. However, the receiving node has to *listen* for prolonged periods of time, and this continuous listening is one of the largest consumers of power in an ad hoc network [158]. Some protocols or applications therefore require nodes to periodically power down their radio interfaces. A node that has powered down its radio interface (and optionally other hardware as well) is said to be in a sleep state. This metric measures the percentage of nodes that can be expected to be active at any given time, and the frequency at which nodes go into sleep states.

4.3 Performance Metrics

Performance metrics quantify the actual data delivery performance of a network. The accurate measurement of these metrics is a prerequisite for evaluation of a network's performance or comparison of performance using different routing protocols.

4.3.1 Byte Delivery Ratio

A network's *byte delivery ratio* is defined as the ratio of data bytes originated by all nodes to data bytes received as final destination by all nodes in the network, excluding data bytes that originated outside the network boundaries.

4.3.2 Routing Overhead Ratio

A network's *routing overhead ratio* is defined as the ratio of control bits originated or forwarded by all nodes in the network to data bits originated or forwarded by all nodes in the network, measured at the network layer.

4.3.3 End-to-end Delay

A network's *end-to-end delay* is defined as the average time interval between the generation and successful delivery of data packets for all nodes in the network, during a given period of time. Packets that are discarded or lost are not included in the calculation of this metric.

4.3.4 End-to-end Throughput

A network's *end-to-end throughput* is a measure of the network's successful transmission rate, and is usually defined as the number of data packets successfully delivered to their final destination per unit of time. However, to convert this metric to a measure of *data throughput* or to compare it to other networks, the network's packet size and the network's number of nodes also has to be known. This work therefore defines a network's end-to-end throughput as the number of data *bytes* successfully delivered to their final destination per unit of time, divided by the number of nodes in the network.

4.3.5 Route Acquisition Time

A network's *route acquisition time* is defined as the average time required for any node to acquire a valid route to an unknown node.

4.3.6 Average power expended

A node's *average power expended* is defined as the average power spent by the node per time period on network-related activities.

4.3.7 Route optimality

A network's *route optimality* is a measure of the optimality of routes used by the network. In traditional fixed-line networks, this is usually the ratio of the shortest available route to the route actually taken by any data packet. However, in ad hoc networks factors such as power consumption and congestion avoidance have to be taken into consideration in addition to route length. Since this is a multi-objective optimisation problem, determination of the optimal route requires a measure of the route's Pareto optimality [159]. Pareto optimality of routes is not covered by this work, and consequently this work does not employ route optimality as a performance metric.

4.4 Qualitative Protocol Features

Qualitative protocol features are distinct characteristics that can be assigned to a protocol, independent of the protocol's implementation. When considering protocols for implementation in a network, these features are often specified at a high level by network designers as required or recommended features.

4.4.1 Knowledge of node locations

Does the routing algorithm require global or local knowledge of node locations? This indicates the amount of information that has to be disseminated to nodes, and is therefore an indication of the network's routing overhead.

4.4.2 Response to topology changes

Does the routing algorithm require incremental updates or complete restructuring after topology changes? This again indicates the amount of information that has to be disseminated to nodes.

4.4.3 Adaptation to radio communication environment

Does the routing algorithm consider radio-link quality, signal strength or noise ratios in routing decisions? An algorithm which does take these factors into account is likely to be more successful in noisy environments.

4.4.4 Power consciousness

Does the routing algorithm consider remaining battery power at intermediate nodes in routing decisions? This is especially critical if the network nodes are battery-powered devices.

4.4.5 Single or multichannel

Does the routing algorithm employ different radio channels for control and data packets? Algorithms that employ different channels are likely to be less sensitive to physical layer packet collisions, and can therefore probably support higher network loads.

4.4.6 Unidirectional or bidirectional links

Does the routing algorithm always assume bidirectional links, or can it efficiently handle unidirectional links? Radio interference can cause wireless links to temporarily become unidirectional, and it is important to establish if an algorithm will be able to cope with these real-life conditions.

4.4.7 Priority message handling

Does the routing algorithm support expediting of high-priority messages, and can the algorithm reduce the delay time of priority messages under high network load conditions? This is especially critical in networks that are used in critical life-support or military environments.

4.5 Quantitative Protocol Features

There are several characteristic features common to all ad hoc networks, such as the time required for new nodes to be integrated into the network. Since each ad hoc network necessarily possesses these features, they are prime candidates for protocol comparisons and for portraying first impressions of the protocol. These features may also determine a protocol's suitability in specialist environments, for instance in an application where fast integration of new nodes is a requirement.

4.5.1 Network Settling Time

A network's *settling time* is defined as the time required from starting up (i.e. switching on) a new network until the first data packet transmitted by each node is successfully received at its final destination.

4.5.2 Network Join Time

A network's *join time* is defined as the time required for the first data packets generated by newly introduced network nodes to be successfully received at their final destinations.

4.5.3 Network Depart Time

A network's *depart time* is defined as the time required for a network to reorganise itself and for each node to successfully transmit the first data packet to its final destination, after the removal (or failure) of a number of network nodes.

4.5.4 Memory Byte Requirement

A routing protocol's *memory byte requirement* is defined as the maximum number of bytes that have to be stored at each node to perform routing activities. This is usually expressed in bytes per node in the network.

4.5.5 Network Scalability Number

A network's *scalability number* is defined as the maximum number of nodes that can be present in the network while maintaining routing of packets to within a predefined set of network performance constraints.

4.6 Conclusion

This chapter provided an overview of evaluation metrics and features in ad hoc networks, specifically for evaluation of ad hoc routing protocols. The focus of this work is the development of an ad hoc routing algorithm, and the success or failure of the algorithm is determined using metrics and features introduced in this chapter.

The chapter started with metrics used to determine the background or environment in which the network is operating. The performance of different networks or routing algorithms can only be weighed against each other if the measurements were made in similar environments. It is therefore of critical importance that any performance publication be accompanied by a full set of these scenario metrics. The chapter then moved on to introduce the specific performance measures relevant to ad hoc networks, and specifically to ad hoc routing protocols.

Most routing algorithms offer some form of a tradeoff in characteristics – for instance, a protocol may offer very quick and reliable packet delivery at the cost of increased overhead. It is therefore important that a comprehensive set of performance metrics for a protocol be available, so that network designers are cognizant of the tradeoffs implicit in each protocol.

Lastly, the chapter described certain characteristics or features that are universal to all ad hoc routing algorithms. As more and more protocols are characterised according to a standard set of characteristics, it is hoped that the information will become increasingly useful to persons or entities considering the real-world implementation of ad hoc networks.

5 Towards better routing

Previous chapters in this work introduced ad hoc networks, algorithms used to route packets in these networks (including algorithms based on the ACO metaheuristic) and metrics used to measure the performance of these algorithms. This chapter aims to evaluate the performance of various sub-mechanisms of these routing algorithms (such as route request, route reply and next-hop decision mechanisms) in conditions as close to real-world as possible. Where shortcomings are identified, improvements are suggested and the impact of such improvements experimentally evaluated.

5.1 Introduction

The diversity of the routing algorithms described in earlier chapters of this work reflects the diversity of the ad hoc networking problem. Although both classified as ad hoc networks, the requirements for an ad hoc network of PDA devices offering multimedia services radically differs from the requirements for an ad hoc network of sensors gathering telemetry data.

However, many algorithms employ common sub-mechanisms in the routing process. Almost all algorithms use a variation of route request or forward ant packets to find destinations, with route reply or backward ant packets completing the route. Almost all algorithms send some form of a route error packet to notify other nodes of link failures. When a node is unable to deliver a packet to a next-hop node, many algorithms employ some form of *backtracking* where the packet is returned to an up-stream node in the hope that the upstream node can successfully forward the packet.

This chapter will isolate and investigate the performance of several of these mechanisms, in conditions that are as close to real-world as possible. To that end, all tests and experiments are performed using reasonably realistic implementations of the media access control (MAC) and physical (PHY) layers. Simulations are performed using version 3.7 of the QualNet network simulator from Scalable Network Technologies. The simulator is supplied with a few standard implementations of ad hoc routing protocols, including AODV.

The 802.11 Distributed Coordination Function (DCF) Media Access Control (MAC) Protocol was chosen for the simulations. Although it has been noted [160] that the 802.11 MAC layer is not an optimum choice for ad hoc networks, it was chosen for this work to allow fair comparisons with other ad hoc routing related works. Signals are transmitted at 2Mb/s on a 2.4GHz carrier frequency. The two-ray propagation path-loss model without fading is used.

Mobility is simulated using the random-waypoint model. Network traffic in all experiments is generated by Constant Bit rate (CBR) sources. The number of CBR sources used is sometimes called *connections* in other works. Where the impact of improvements on a routing algorithm needs to be experimentally verified, the experiments are performed on standard and modified versions of the AODV routing algorithm.

The focus of this work is on network-layer routing activities. As such, the impact of the 802.11 MAC layer on routing activities will be shown, but the MAC layer itself will not be discussed in detail. The interested reader is referred to [161]. The only detail regarding the 802.11 MAC layer that is deemed to be relevant to routing activities (for the purpose of this work) is the 4-step packet handoff process shown in Figure 14.

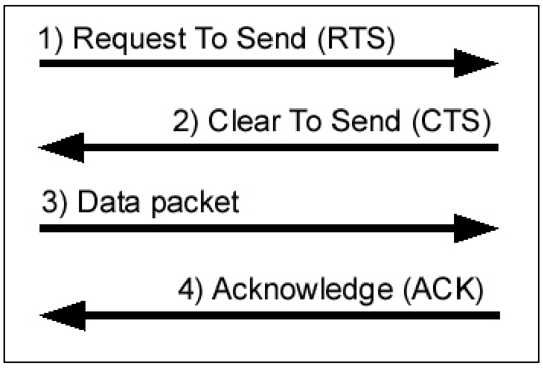


Figure 14 : 802.11 MAC layer unicast handoff sequence

As a last note on the 802.11 MAC layer, it is interesting to note that it is possible for a node using this MAC layer to receive the same data packet *twice.* After transmitting the data packet (step 3 in Figure 14) the source expects an ACK packet from the destination. If the destination transmits an ACK packet but it is not successfully received by the source, then the destination will see the transmission sequence as completed and will have received a valid data packet. The source waits a predetermined time for the ACK packet, upon not receiving the ACK packet marks the transmission sequence as failed, and may retry the entire 4-step transmission sequence causing the destination to receive the same data packet again. In some experiments carried out for this work where data packets were not checked for uniqueness, this curiously resulted in the simulator reporting byte delivery ratios of more than 100%.

Unless otherwise stated in the text, all experiments performed in this chapter used the general network setup summarised in Table 4. Most experiments were repeated 20 times with different random seed values and different node starting positions, but with the application layer kept constant for a given set of experiments.

Table 4: General experimental setup used in this chapter

Parameter	***Value***
Number of nodes in network	100
Node initial placement	Random
Number of CBR sources	5
Number of data packets created by each CBR source	1000
Data packet size	1024 bytes
Terrain size	3000m x 1000m
CBR source start time	Random up to 10 seconds
Random Waypoint pause time (s)	1
Number of times experiment repeated	20

The rest of this chapter is organised as follows. Section 5.2 defines confidence intervals to be used in certain experimental results. The routing overhead ratio is clarified in section 5.3, and the backtracking mechanism is evaluated in section 5.4. Sections 5.5 and 5.6 look at the RREQ and RREP mechanisms respectively. The mechanism used to make next-hop decisions is examined in section 5.7, and section 5.8 concludes this chapter.

5.2 Confidence Intervals

Where appropriate, experimental results are presented with 95% confidence intervals and standard deviations. In all cases, 95% confidence intervals are calculated using [162]:

$$\bar{x} \pm 1.96\left(\frac{\sigma}{\sqrt{n}}\right) \tag{5.1}$$

where $\bar{x}$ is the arithmetic mean of the samples, n is the number of samples, and σ is the standard deviation of the samples calculated as [162]:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{(n-1)}} \tag{5.2}$$

where $\bar{x}$ is the arithmetic mean of the samples, n is the number of samples, and x_i is the value of the i^{th} sample.

5.3 Clarifying the Routing Overhead Ratio metric

Nearly every work on ad hoc networks uses the network's *routing overhead ratio* as a performance metric. Routing overhead ratio is a measure of a routing protocol's efficiency, and is especially critical in networks where efficient use of battery power or network bandwidth is required. Yet there does not appear to be a universal understanding of what the routing overhead ratio is. Table 5 lists a few definitions of this metric in various works.

Table 5: Definitions of routing overhead ratio in different works

Definition of Routing Overhead Ratio	***Reference***
… the number of routing packets "transmitted" per data packet "delivered" at the destination. Each hop-wise transmission of a routing packet is counted as one transmission.	[52]
… the fraction of routing packets needed to deliver a data packet. We counted bits used for routing, because the different protocols generate the overhead in very different ways.	[147]
… the network load generated by the routing packets is reported as the ratio between the bandwidth occupied by the routing packets and the total available network bandwidth.	[138]
Normalized routing overhead is the number of routing packets transmitted per data packet received at the destination.	[150]
The average number of control packets produced per mobile node.	[163]
For a node n, this is the sum of duplicated packets received by n.	[164]
The number of control packets for the routing protocol over the number of data packets sent.	[165]

The definitions listed above are unacceptably diverse, and makes comparison of different works virtually impossible. Moreover, works that define a network's routing overhead ratio in terms of the number of packets produce inaccurate results, as the effects of different control packet sizes are not taken into account. This is especially evident when distance-vector routing protocols (which generate constantly sized control packets) are compared with source routing protocols which generate constantly *growing* control packets. Assuming 4-byte IP addresses, Figure 15 illustrates the differences in control packet sizes between AODV and DSR – these packets can clearly not all be treated as equal.

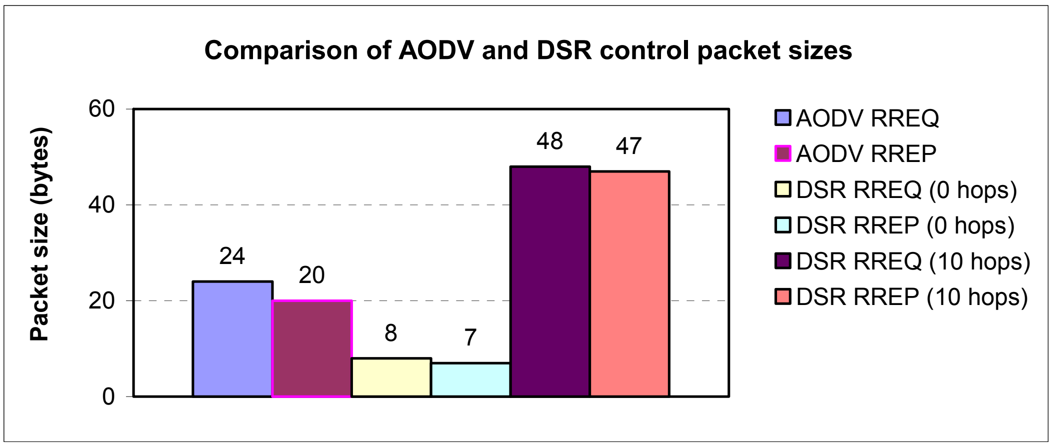


Figure 15 : Comparison of AODV and DSR control packet sizes

This work endeavors to move towards standardisation of ad hoc nomenclature, and therefore suggests the following definition of routing overhead ratio (from section 4.3.2):

"A network's routing overhead ratio is defined as the ratio of control bits originated or forwarded by all nodes in the network to data bits originated or forwarded by all nodes in the network, measured at the network layer."

Although a step in the right direction, the above definition still leaves much room for interpretation. Consider a routing algorithm that creates multiple routes to destinations. If node s attempts to forward a data packet to destination d through next-hop n_1 and the transmission fails, s will retry the transmission through next-hop nodes $n_2, n_3, \ldots, n_i$ assuming that s has i routes to d available. If the i^{th} transmission succeeds, s will have *delivered* one packet of data, but *initiated* and spent power and bandwidth resources on i packets of data. It can certainly be argued that a perfect algorithm would not have wasted resources on the failed transmissions and that an algorithm should be penalised (in terms of routing overhead) for such transmissions.

Now consider the case where all i transmissions from node s fail, and the algorithm backtracks the data packet to an upstream node. The packet is by definition a *data* packet, and in terms of the routing overhead ratio definition should be recorded on the data side of the routing overhead ratio. However the transmission of the packet to the upstream node has not brought the packet closer to destination d. It can again be argued that a perfect algorithm would have avoided the dead-end at node d which necessitated backtracking, and that such transmissions should be recorded as overhead bytes.

The definition of routing overhead ratio given above is therefore extended to the following two interpretations:

- **Strict routing overhead**, in which bits are only recorded as data bits when they form part of a data packet that was successfully transmitted on a forward route towards a destination; and
- **Standard routing overhead,** in which any bits that form part of any initiated data packet are recorded as data bits.

To illustrate the effect that these two interpretations have on reported routing overhead ratio, a set of experiments was performed where the routing overhead ratio was measured according to both interpretations. The AODV routing protocol was used for packet routing, but modified so that routes are allowed to keep multiple routes to a destination. Table 6 presents an overview of the experimental parameters.

Table 6: Experimental setup to compare routing overhead ratio interpretations

Parameter	***Value***
CBR source packet generation rate	Random from 1 to 3 seconds
Random Waypoint maximum speed (m/s)	Stepped from 0m/s to 100m/s in 10m/s increments
General	Route reply by intermediate nodes enabled Backtracking of data packets enabled

Experiments were repeated 20 times with different random seed values. The averaged results and 95% confidence intervals are shown in Figure 16 and Table 7. At very low node mobility, the two interpretations yield roughly similar results. This is as a result of few packets being retransmitted or backtracked, since link failure occurs less in low mobility networks. The difference between the two results increase sharply as node mobility is increased. With mobility increasing from 0 to 20 m/s, the standard routing overhead ratio only rises to 0.13, while the strict routing overhead ratio reaches 0.50 – a significant difference.

Interestingly, the standard routing overhead appears to stabilize at higher node mobility, while the strict routing overhead curve flattens much slower. The standard routing overhead only rises by 23% from 20 to 100 m/s, while the strict routing overhead rises by 44% in the same interval. This indicates that increased data packet retransmission and backtracking, which is interpreted as data in the standard routing overhead ratio, is offsetting increased numbers of RREQ/RREP/RERR packets at higher mobility, so that the resulting ratio using the standard routing overhead ratio stays more-or-less constant.

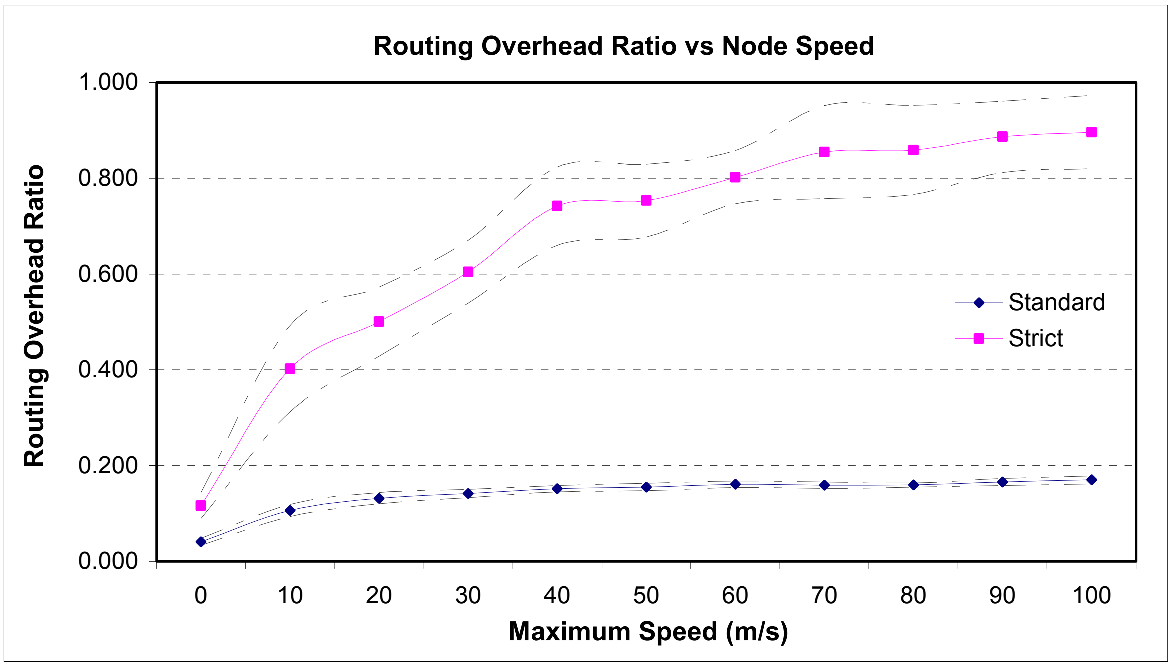


Figure 16 : Comparison of routing overhead ratio with different interpretations

Table 7: Comparison of routing overhead ratio with different interpretations

	Standard			*Strict*		
Max Speed	***Overhead Ratio***		***σ***	***Overhead Ratio***		***σ***
0	0.041	+- 0.007	0.012	0.116	+- 0.027	0.044
10	0.106	+- 0.013	0.021	0.402	+- 0.090	0.145
20	0.132	+- 0.012	0.019	0.500	+- 0.072	0.116
30	0.141	+- 0.009	0.014	0.605	+- 0.065	0.106
40	0.151	+- 0.007	0.011	0.742	+- 0.082	0.132
50	0.155	+- 0.008	0.012	0.753	+- 0.076	0.123
60	0.161	+- 0.007	0.011	0.802	+- 0.056	0.090
70	0.159	+- 0.007	0.011	0.854	+- 0.097	0.157
80	0.159	+- 0.004	0.007	0.859	+- 0.093	0.150
90	0.165	+- 0.007	0.012	0.886	+- 0.074	0.120
100	0.170	+- 0.008	0.014	0.896	+- 0.076	0.123

This work views the strict routing overhead ratio as a more accurate indication of a protocol's efficiency at delivering data packets to their destinations, and therefore all experimental results reported in this work use the strict definition of routing overhead ratio.

5.4 Backtracking of data packets

When a node's efforts to forward a data packet to the packet's destination fails, some algorithms [147, 166] employ a *backtracking* mechanism where the data packet is sent to an upstream node for further routing. Although backtracking may improve an algorithm's packet delivery ratio, it could have serious routing overhead and packet delay implications as data packets are sent back-and-forth along several routes. The purpose of this section is to experimentally evaluate the implications of backtracking.

When a packet is backtracked back to its source node, the node may reinitiate a RREQ procedure or drop the packet. In the simulations, such packets were dropped. To evaluate the performance of backtracking, two sets of experiments were performed with backtracking enabled in the first set and disabled in the second set. Table 8 presents an overview of the experimental parameters.

Table 8: Experimental setup to evaluate backtracking

Parameter	***Value***
CBR source packet generation rate	Random from 1 to 3 seconds
Random Waypoint maximum speed (m/s)	Stepped from 0m/s to 100m/s in 10m/s increments
General	Route reply by intermediate nodes enabled

Experiments were repeated 20 times with different random seed values. The averaged byte delivery ratio and 95% confidence intervals are shown in Figure 17 and Table 9. At low node mobility, backtracking has very little effect as link failures occur infrequently and backtracking is usually not necessary. As can be expected, backtracking improves the byte delivery ratio of the algorithm at higher node mobility. The improvement in byte delivery ratio increases with increasing node mobility, as more link failures occur, with a maximum improvement of 10.2% in the observed mobility range.

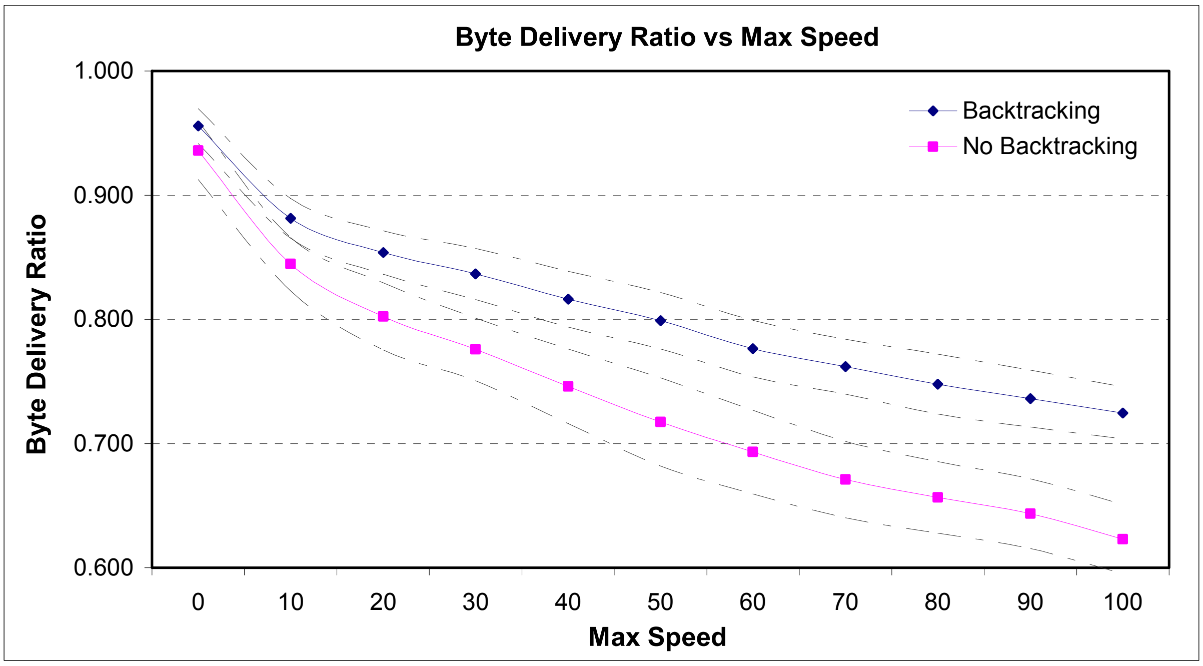


Figure 17 : The impact of backtracking on byte delivery ratio

Table 9 : The impact of backtracking on byte delivery ratio

	Backtracking			***No Backtracking***		
Max Speed	***Delivery Ratio***		***σ***	***Delivery Ratio***		***σ***
0	0.956	+- 0.014	0.022	0.936	+- 0.023	0.038
10	0.881	+- 0.016	0.026	0.844	+- 0.021	0.035
20	0.854	+- 0.017	0.028	0.802	+- 0.027	0.044
30	0.837	+- 0.021	0.033	0.776	+- 0.025	0.041
40	0.816	+- 0.022	0.036	0.746	+- 0.030	0.048
50	0.799	+- 0.023	0.037	0.717	+- 0.036	0.057
60	0.776	+- 0.023	0.037	0.693	+- 0.034	0.054
70	0.762	+- 0.022	0.036	0.671	+- 0.031	0.050
80	0.748	+- 0.024	0.039	0.657	+- 0.029	0.047
90	0.736	+- 0.023	0.037	0.644	+- 0.028	0.045
100	0.725	+- 0.021	0.034	0.623	+- 0.028	0.045

The cost at which the increased byte delivery ratio is achieved is depicted Figure 18 and Figure 19, with Table 10 and Table 11 showing the standard deviation and 95% confidence intervals. Both sets of curves follow similar (though inverted) trends than the byte delivery ratio, starting at comparable values at low node mobility, and showing increasing divergence at higher node mobility. The general trends of these curves are to be expected, as the increased byte delivery ratio at higher node mobility is achieved by generating more routing overhead (backtracked data packets), and backtracked packets will certainly take longer to arrive at their destinations than non-backtracked packets.

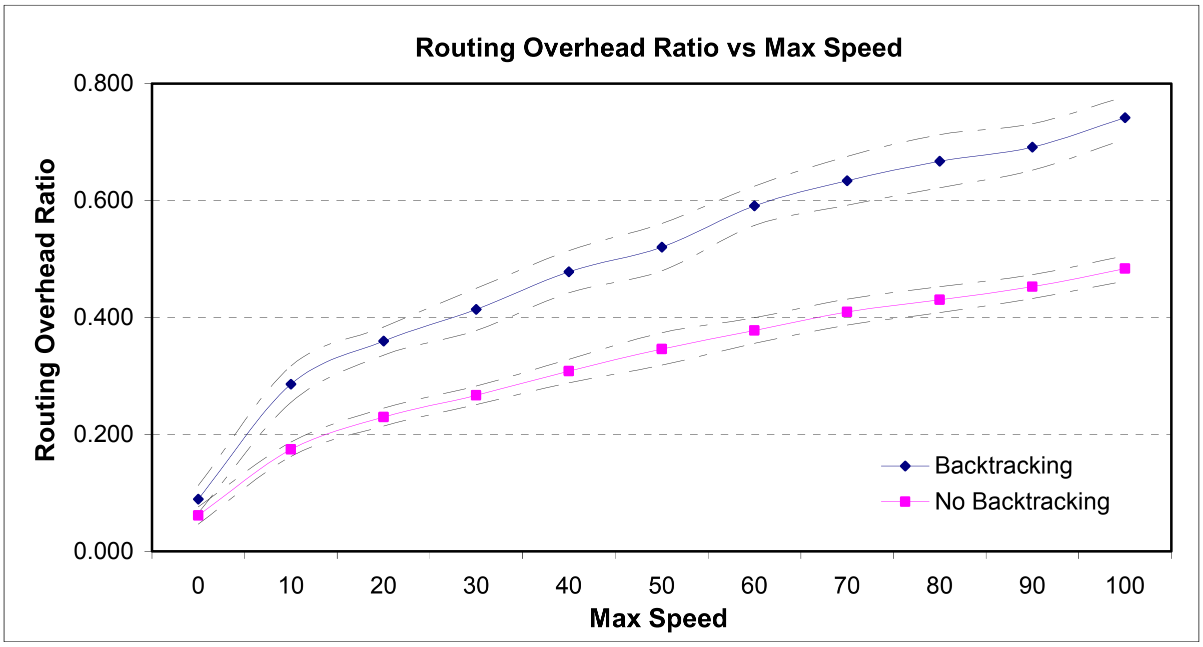


Figure 18 : The impact of backtracking on routing overhead ratio

Table 10 : The impact of backtracking on routing overhead ratio

	Backtracking			***No Backtracking***		
Max Speed	***Overhead Ratio***		***σ***	***Overhead Ratio***		***σ***
0	0.089	+- 0.024	0.038	0.061	+- 0.015	0.024
10	0.286	+- 0.031	0.050	0.175	+- 0.012	0.020
20	0.360	+- 0.024	0.039	0.229	+- 0.015	0.025
30	0.414	+- 0.036	0.058	0.267	+- 0.016	0.026
40	0.478	+- 0.036	0.058	0.308	+- 0.020	0.033
50	0.520	+- 0.041	0.065	0.346	+- 0.028	0.044
60	0.591	+- 0.034	0.054	0.378	+- 0.022	0.035
70	0.634	+- 0.042	0.067	0.409	+- 0.022	0.036
80	0.667	+- 0.045	0.073	0.430	+- 0.022	0.035
90	0.692	+- 0.040	0.064	0.453	+- 0.021	0.033
100	0.741	+- 0.036	0.059	0.484	+- 0.022	0.035

The maximum increase in byte delivery ratio from 62.3% to 72.5% at 100m/s represents a 16.3% improvement. This improvement was realized at the cost of a 53.3% increase (from 48.4% to 74.1%) in routing overhead and a 23.8% increase (from 0.074s to 0.092s) in packet end-to-end delay. In other words, backtracking costs much more in terms of routing overhead than in terms of end-to-end delay.

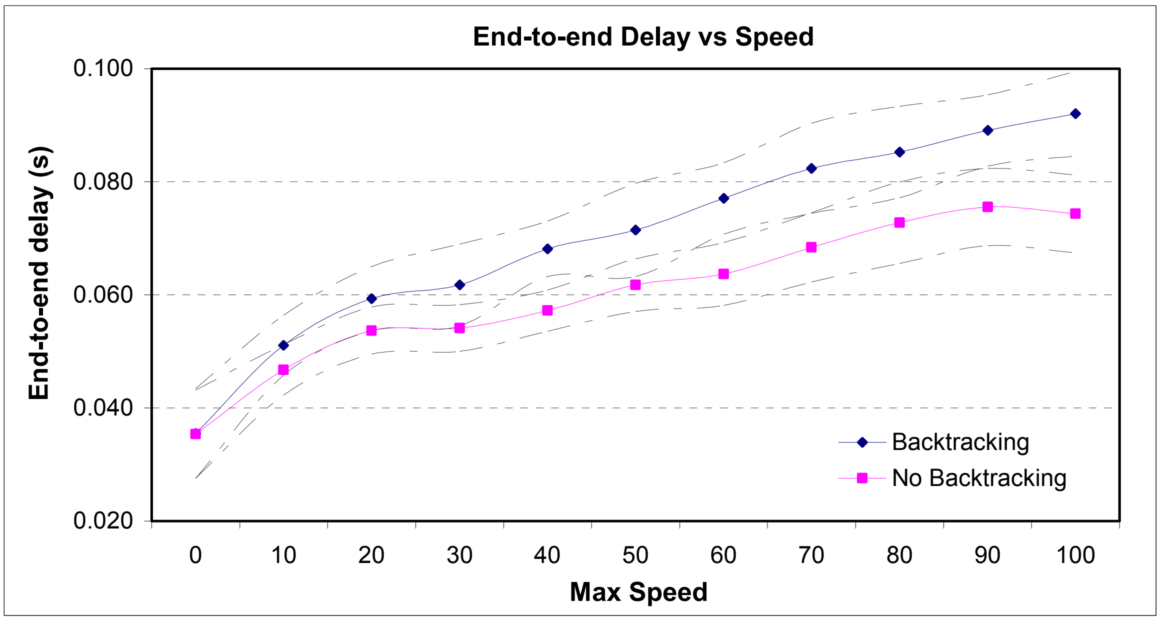


Figure 19 : The impact of backtracking on packet end-to-end delay

Table 11 : The impact of backtracking on packet end-to-end delay

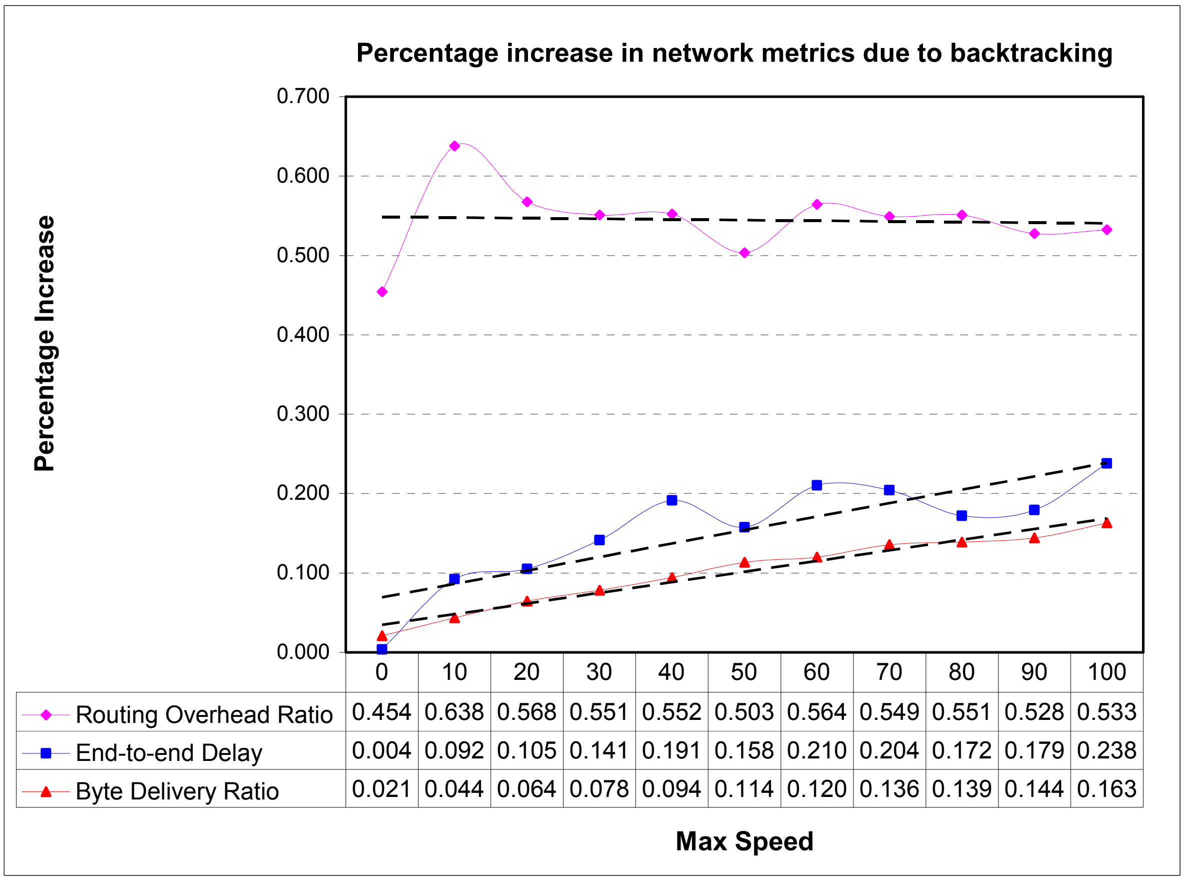
	Backtracking			*No Backtracking*		
Max Speed	*End-to-end Delay*		*σ*	*End-to-end Delay*		*σ*
0	0.035	+- 0.008	0.013	0.035	+- 0.008	0.013
10	0.051	+- 0.005	0.009	0.047	+- 0.004	0.007
20	0.059	+- 0.006	0.009	0.054	+- 0.004	0.007
30	0.062	+- 0.007	0.012	0.054	+- 0.004	0.007
40	0.068	+- 0.005	0.008	0.057	+- 0.004	0.006
50	0.071	+- 0.008	0.013	0.062	+- 0.005	0.008
60	0.077	+- 0.006	0.010	0.064	+- 0.006	0.009
70	0.082	+- 0.008	0.013	0.068	+- 0.006	0.010
80	0.085	+- 0.008	0.013	0.073	+- 0.007	0.012
90	0.089	+- 0.006	0.010	0.076	+- 0.007	0.011
100	0.092	+- 0.007	0.012	0.074	+- 0.007	0.011

In the experimental network setup, all data packets are constantly sized and the minimum time taken by a node to forward a data packet (excluding time spent in packet queues) is a constant (usually called the *node traversal time*). Route repair by intermediate nodes on a route is disabled, so an intermediate node can only forward or backtrack a packet.

Consider a network where data packets are always processed within this minimum time, i.e. no time is spent in packet queues. Each time the backtracking mechanism is used, the packet is sent to an upstream node generating a constant number of overhead bytes, and delayed by a constant time. In such a network, the increase in routing overhead ratio due to backtracking will therefore be directly proportional to the increase in end-to-end packet delay due to backtracking across the observed range of node mobility.

Figure 20 depicts the percentage increases of routing overhead ratio, end-to-end delay and byte delivery ratio over the observed mobility range. To aid visualization of graph trends, trend lines were added to the various series using least-squares linear regression.

The percentage increases in routing overhead ratio and end-to-end delay is clearly not proportional. This indicates that, when using backtracking, the component of end-to-end delay caused by packets waiting in packet queues increases with increasing node mobility. It can also be observed that the routing overhead penalty paid for backtracking remains fairly constant over the mobility range. In the observed network, the maximum and minimum increases in overhead were 63.8% and 45.4% respectively. For node mobility above 20 m/s, backtracking caused a roughly constant increase of around 55% in routing overhead ratio.



	0	10	20	30	40	50	60	70	80	90	100
Routing Overhead Ratio	0.454	0.638	0.568	0.551	0.552	0.503	0.564	0.549	0.551	0.528	0.533
End-to-end Delay	0.004	0.092	0.105	0.141	0.191	0.158	0.210	0.204	0.172	0.179	0.238
Byte Delivery Ratio	0.021	0.044	0.064	0.078	0.094	0.114	0.120	0.136	0.139	0.144	0.163

Figure 20 : Percentage increase in routing metrics due to backtracking

In summary, the backtracking mechanism observed in this section displayed the following characteristics:

- Backtracking provided byte delivery ratio improvements of 2.1% (0m/s), up to 16.3% (100m/s),
- Backtracking increased the network's routing overhead ratio on average by a constant of roughly 55%, and
- Backtracking increased the network's end-to-end delay by 0.4% (0m/s) up to 23.8% (100m/s).

5.5 Sending RREQ packets

Routing algorithms that maintain multiple routes between a source and destination node have an inherent advantage over algorithms that maintain a single exclusive route – when a route fails, the node may have a backup route readily available, and packet transmission may continue undisturbed. The usual mechanism employed to discover multiple routes are the sending of multiple packets by the source that traverse the network in search of routes to the destination.

In ACO nomenclature these packets are usually called *forward ants*, in various other ad hoc related works the packets are called *route request (RREQ)* packets. Forward ants can be seen as a special case of RREQ packets that specifically employ pheromone to create routes back to the source. In an effort to standardise naming conventions, this work will refer to these packets generically as RREQ packets.

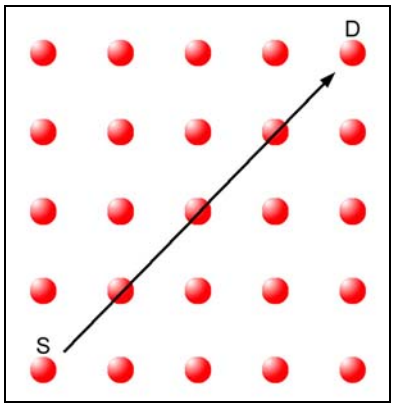


Figure 21 : Network setup used for testing of RREQ propagation.

Consider the 25-node network setup in Figure 21. The source node (node **S**) requires routes to the destination node (node **D**) to forward data packets. *Broadcast jitter* in networks refers to a small random delay applied before sending any data packet in an effort to reduce packet collisions, and is usually specified as the upper bound of a random delay function [167]. Figure 22 shows the results of sending a single RREQ packet to the destination, repeated over 4 independent experiments, each time setting the broadcast jitter to 10 milliseconds. The sequence number mechanism of the AODV routing algorithm ensures that each node will only rebroadcast the packet once, so that the packet's transmission over the network effectively forms a spanning tree rooted at the source node. In all four experiments the RREQ packet reaches the destination, indicating that a route from the source to the destination was found. It can also be noted that the RREQ packets covered on average 97% of the network, and that there appears to be reasonable diversity between the routes discovered in the 4 cases.

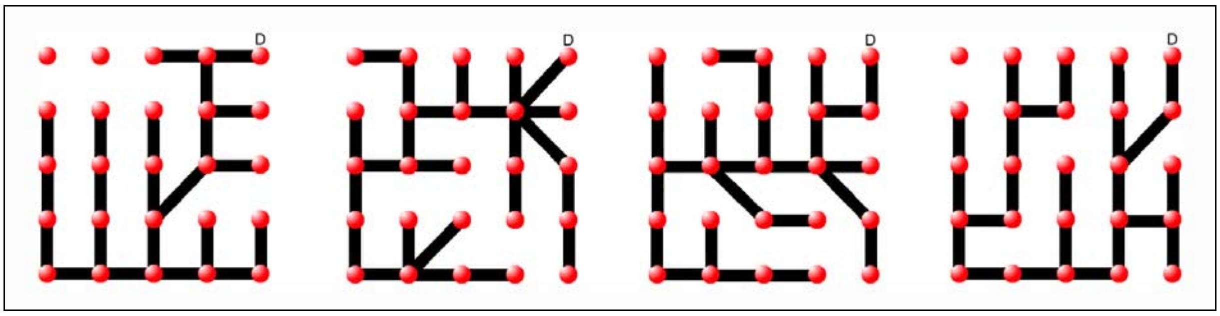


Figure 22 : Propagation of a single RREQ packet through a 25-node network

The experiment was now repeated with 4 RREQ packets being sent in a single experiment, to simulate a node generating multiple RREQ packets in search of multiple routes to the destination. The experiment was repeated four times with broadcast jitter set to 1, 10, 100 and 500 milliseconds in the four respective experiments, and in each experiment the path of each of the four RREQ packets was recorded as they propagated through the network.

Figure 23 shows the results of the experiments. In the first experiment with broadcast jitter set to 1 millisecond, *none* of the RREQ packets reached the destination, and the RREQ packets covered on average only 22% of the network. The third RREQ packet sent in this experiment was not received by a single neighbouring node. It seems counter-intuitive that *more* RREQ packets will cover *less* of the network.

Analysis of the experimental results shows that the packets were lost due to MAC layer collisions. This can also be observed from the fact that as the broadcast jitter is increased and the chances of MAC layer collisions decrease, each of the RREQ packets cover more of the network and the destination node is reached more often. Figure 24 shows the percentage of the network covered and percentage of times the destination was reached for the four experiments. In both the 100ms and 500ms experiments the destination is reached 75% of the time, but the packets sent with 100ms broadcast jitter only covered 88% of the network compared to 94% of the network covered by packets sent with 500ms broadcast. As the broadcast jitter is increased, both the percentage of packets reaching the destination and the percentage of the network covered, approaches the results obtained in the first set of experiments with four independent RREQ packets.

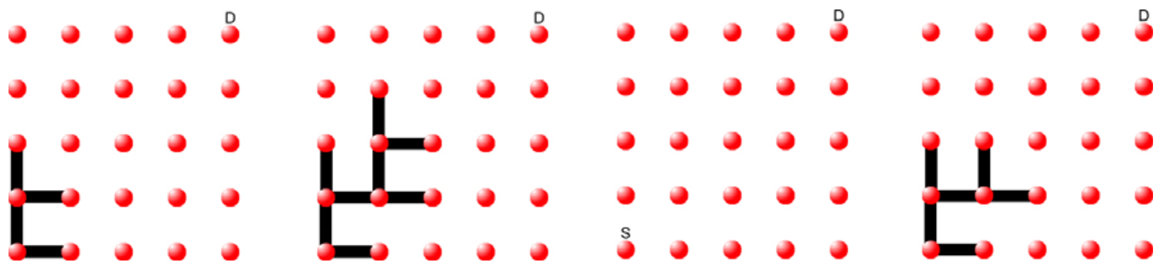
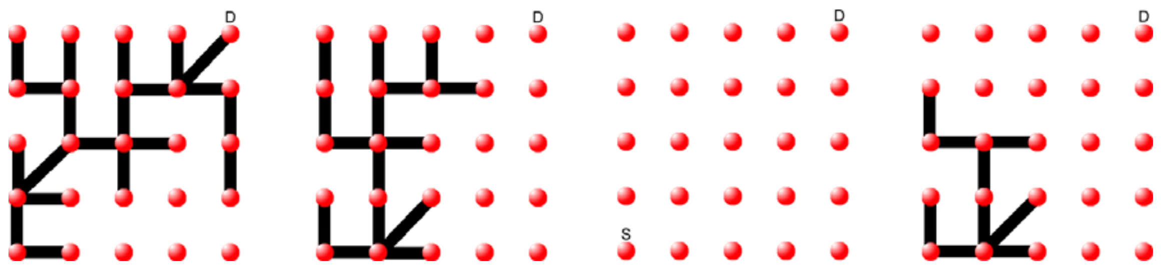
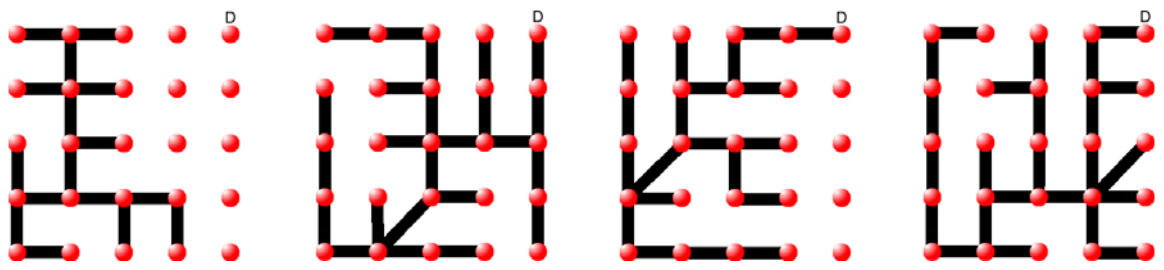
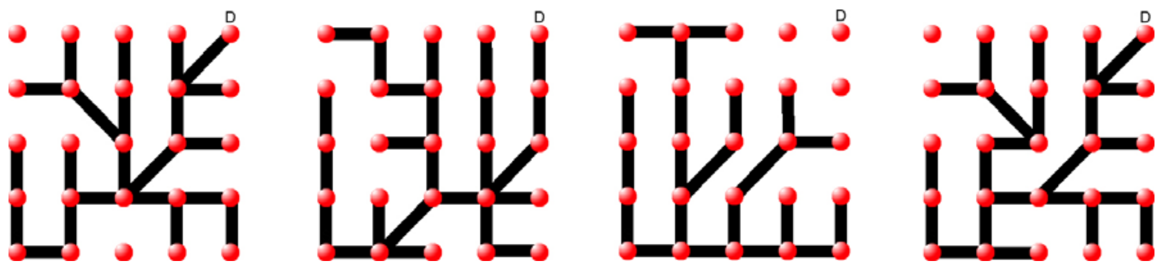
Jitter (ms)	*Propagation of RREQ packet*
1	
10	
100	
500	

Figure 23 : Propagation of four successive RREQ packets

The routes (or partial routes) discovered with broadcast jitter set at 1 and 10ms are also less diverse than the routes discovered in the first set of independent experiments. As the broadcast jitter is increased, the routes become more diverse.

In experiments using a perfect MAC layer packets are delivered from one node to the next if the receiving node is within the calculated radio range of the transmitting node [148], and this phenomenon will therefore not be observed in such experiments. This phenomenon will also be less visible in networks that generate single RREQ packets at predefined intervals if the intervals are long enough to allow packets to be broadcast without collisions.

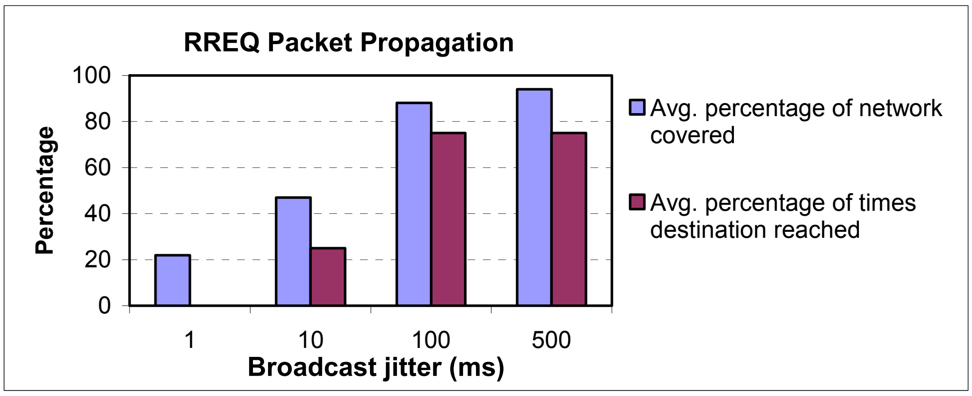


Figure 24 : Impact of broadcast jitter on RREQ propagation

These experiments show that ad hoc nodes sending multiple RREQ packets should allow sufficient time between transmissions (either through scheduled transmissions or broadcast jitter) to avoid packet collisions. Some networks proactively send out RREQ packets over a period of time, thereby eliminating the need for large broadcast jitter. However, in many applications a node in a sensor network may lie dormant in the field for months without sending or receiving data. Such networks only send RREQ packets on-demand, and therefore need to allow sufficient broadcast jitter between RREQ transmissions. Excessive broadcast jitter could lead to long end-to-end data packet delays or routes that became stale during the jitter delay period. The broadcast jitter therefore has to be tuned to each application's needs.

5.6 Sending RREP packets

As RREQ packets propagate through the network they create routes from the current node visited by the packet back to the source. These routes can be used by any packet traveling from the *destination* to the *source,* but routes from the *source* to the *destination* – the actual routes that will be used by the source to send data packets – are still to be created. When a RREQ packet reaches the destination, the destination node usually creates a packet to send back to the source, thereby creating the route that will be used by data packets traveling from the source to the destination.

In ACO nomenclature these packets are usually called *backward ants*, in various other ad hoc related works the packets are called *route reply (RREP)* packets. Backward ants can be seen as a special case of RREP packets that specifically employ pheromone to create routes back to the destination. In an effort to standardise naming conventions, this work will refer to these packets generically as RREP packets.

To study the behaviour of RREP packets traveling from the destination back to the source, a *Route Reply Delivery Ratio* metric is defined as follows:

$$\text{Route Reply Delivery Ratio} = \frac{\begin{array}{c}\text{Total Number of RREP packets received}\\ \text{as source networkwide}\end{array}}{\text{Total Number of RREP packets initiated networkwide}} \tag{5.3}$$

In order to isolate the behaviour of RREP packets originating from destination nodes, RREP by intermediate nodes are disabled in all the following experiments, so that the route reply delivery ratio metric will effectively express the percentage of RREP packets that successfully traveled from the destination node back to the source node.

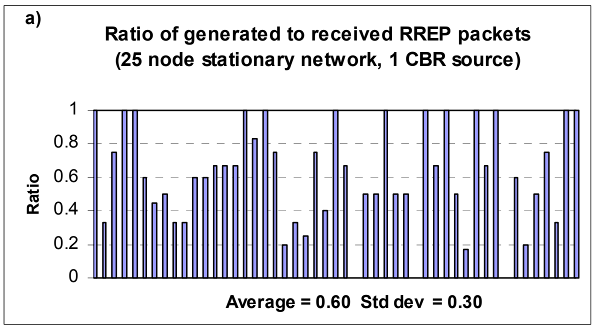
Table 12: 25-node experimental setup to test RREP propagation

Parameter	***Run 1***	***Run 2***	***Run 3***	***Run 4***
Number of nodes in network	25			
Number of CBR sources	1	1	25	25
Number of data packets created by each CBR source	1			
Terrain size	1000m x 1000m			
Experiment time	50 seconds			
CBR source start time	Random up to 10 seconds			
Route reply by intermediate node with valid route	Disabled			
Random Waypoint pause time (s)	0			
Random Waypoint maximum speed (m/s)	0	25	0	25

The same network setup used to test the propagation of RREQ packets (see Figure 21) was now used to test the propagation of RREP packets. In each experiment, a single CBR data source is trying to send a single data packet from node **S** to node **D**. The experiment was first conducted using a stationary network, and then repeated on a network where nodes move according to the random waypoint model with a maximum speed of 20 m/s. In each case, the experiment was repeated 50 times with different random seed values.

A second set of experiments was performed to illustrate the effects of network loading on the propagation of RREP packets. In each of these experiments, 25 separate CBR data sources are each trying to send a single data packet from node **S** to node **D**. Similar to the first set of experiments, these experiments were first conducted on a stationary network and then repeated on a network with moving nodes, and each set of experiments repeated 50 times with different random seed values. A summary of these experimental parameters is given in Table 12.

Figure 25 shows the results of the first two sets of experiments. The network setup of Figure 25a is just about the simplest scenario possible in ad hoc networking – a stationary network, only 25 nodes which are all within range of at least 3 neighbour nodes, and only a single data packet to be routed. Yet on average only 60% of the RREP packets were successfully delivered back to the source. In the AODV routing protocol this translates to a RREQ retry rate of at least 40%, with the associated packet delay and routing overhead implications.



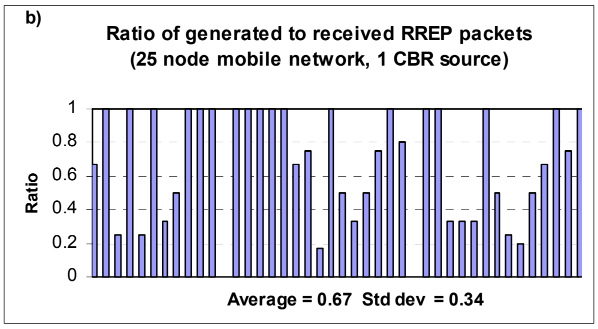
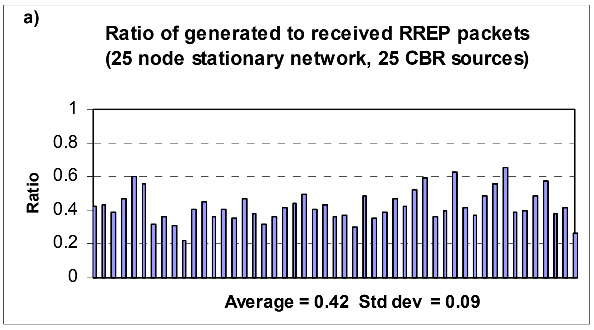


Figure 25 : Route Reply Delivery Ratio using a single CBR source

Analysis of the experimental data shows that RREP packets are dropped when the bi-directional RTS/CTS mechanism of the 802.11 MAC layer (see Figure 14) fails to complete. RREQ packets are broadcast from one node to another without an acknowledgement from the receiver, and are therefore only a testament to the performance of the radio link in one direction. RREP packets retrace the route followed by RREQ packets, but are unicast using the bi-directional RTS/CTS mechanism. Many RREP packets were dropped after receiving no reply at all from the next-hop node. In other cases, the transaction was successfully initiated (RTS sent by sender, CTS sent by receiver) but failed during the Data or ACK cycles. Although a detailed analysis of the 802.11 MAC layer is outside the scope of this work, it can be speculated that this occurs when nodes are close to the borders of radio range, or when packet transmissions were interrupted by interference from surrounding nodes.

The route reply delivery ratio achieved with nodes moving at a maximum speed of 20m/s (shown in Figure 25b) is at 67% average slightly higher than the ratio seen in a stationary network. The higher mobility of the nodes causes the probability of packet collisions to decrease, allowing more RREP packets to reach the source node. This phenomenon was also reported by Johansson *et al* in [168].

Figure 26 shows the results of the second set of experiments using 25 CBR sources. The route reply delivery ratio in Figure 26a have dropped from 60% with 1 CBR source to 42% with 25 CBR sources. This appears to support the notion that packet loss is caused partially by packet collisions. Consistent with the experiments using a single CBR source, the delivery ratio in Figure 26b is at 52% slightly higher (due to higher node mobility) than in Figure 26a. However both route reply delivery ratios in Figure 26 are lower than the ratios in Figure 25, suggesting that the effects of node mobility (at least at these speeds) cannot fully compensate for the packet loss caused by higher network load.



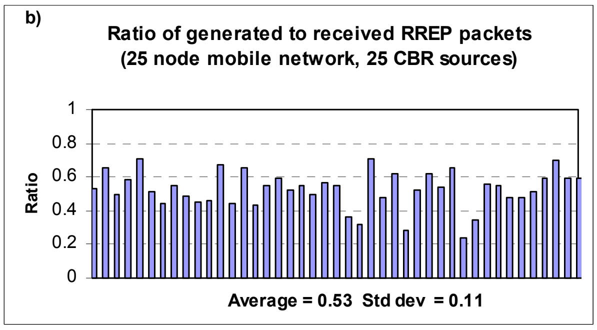


Figure 26 : Route Reply Delivery Ratio using 25 CBR sources

It seems clear from the above experiments that the current RREQ/RREP mechanism is inadequate to reliably provide routes from source to destination nodes. Even in the cases where a RREP packet did make it back to the source node, the RREP packet only represents a single route to the destination. In a single-route algorithm like AODV, this means that a single link failure may force the source node to reinitiate the RREQ process. Multi-route algorithms like Ant-Colony Based Routing (ARA) rely on the RREQ/RREP mechanism to provide multiple routes from which the ACO algorithm can choose attractive next-hop nodes. If the RREQ/RREP mechanism can only provide one or two routes to the destination, the value of the ACO algorithm is greatly reduced.

The experiments performed in section 5.5 indicated that a single RREQ packet is effective at propagating through most of the network, thereby creating routes back to the source at multiple nodes. If a destination node responds to a RREQ packet by broadcasting a RREP back to the source rather than unicasting it, the number of intermediate nodes that have active routes to the destination will be dramatically increased (of course, at the cost of slightly higher network overhead). Lu *et al* [169] note that the unicasting of RREP packets from destination to source prevents valuable routing information from being propagated to other nodes. However, it should be noted from a practical point of view that the unicasting of a packet from destination to source uses the same 4-step 802.11 MAC layer handoff (Figure 14) that will be used to send data packets, and therefore ensures that the entire route consists of viable bi-directional links.

The view taken by this work is that the node best suited to provide routes to the destination is the destination node itself. The allocation of the following two additional bits in the traditional RREQ packet is therefore proposed:

Table 13: Two extra bits added to a RREQ packet

Bit Name	***Description***
ReqRoutingAssistance	Added to a RREQ packet, this bit requests the destination node to assist the source in finding routes from source to destination
SendRREP	Added to a RREQ packet, this notifies receiving nodes whether or not a response to this RREQ packet should be sent

Any node using a multi-route protocol or wishing to secure more routes to the destination may send a RREQ packet with the *ReqRoutingAssistance* bit set. A destination node receiving a RREQ packet with the *ReqRoutingAssistance* bit set will respond by unicasting a RREP packet back to the source. In addition, the destination node will broadcast a number of RREQ packets to seed the network with routes back to itself. The routes discovered by these RREQ packets will not be used by the destination, but only by the source. The destination therefore does not require any nodes to send RREP packets back to it, and will clear the *SendRREP* bit in the sent RREQ packets to notify nodes to simply relay the RREQ packet without sending route replies. Additionally, every time the node receives a data packet as destination there is a small probability p_{dard} (a tunable network parameter) that the node will broadcast a RREQ packet with the *SendRREP* bit cleared, to refresh the routes that intermediate nodes have to the destination. This mechanism is similar to the route seed packets introduced by Roth and Wicker in the Termite algorithm [148]. To simplify further discussion, this mechanism will be referred to as *destination assisted route discovery (DARD)*.

In initial experiments, a protocol using DARD performed *worse* in terms of byte delivery ratio and routing overhead ratio than a network using the standard AODV algorithm. Even when $p_{dard} = 0$, so that a destination node will launch RREQ packets only in response to a RREP packet, performance was worse than the standard AODV algorithm. Experimental data showed that although nodes were receiving RREQ packets and initiating RREP packets, the route reply delivery ratio dropped to less than 0.1 – in other words, more than 90% of the RREP packets were dropped on their way from destination back to source. It appears that the RREQ packets being broadcast simultaneously with the unicasting of RREP packets caused RREP packets to be dropped due to MAC layer collisions. The broadcasting of RREQ packets on receipt of a RREQ packet as final destination was therefore removed from the DARD mechanism, leaving only the broadcasting of RREQ packets with probability p_{dard} on receipt of a data packet as final destination.

To test the effects of DARD, a 100-node experimental setup was created with 5 random CBR source nodes each sending 1000 data packets to a random destination node. Each CBR source generates a data packet at a random interval of between 0.1 and 0.3 seconds, and maximum node mobility was set to a relatively high level of 60m/s. The value of p_{dard} was stepped from 0 (no RREQ packets generated by the destination) to 0.45 in steps of 0.05. A summary of these experimental parameters is given in Table 14.

Table 14: 100-node experimental setup with different p_{dard} values

Parameter	***Value***
Experiment time	400 seconds
CBR source start time	Random up to 10 seconds
CBR source packet generation rate	Random from 0.1 to 0.3 seconds
Random Waypoint maximum speed (m/s)	60m/s
p_{dard}	Stepped from 0.00 to 0.50 in 0.05 steps
General	Route reply by intermediate nodes enabled Backtracking enabled

Experiments were repeated 20 times with different random seed values. Figure 27 and Table 15 show the byte delivery ratio as a function of p_{dard} with standard deviation and 95% confidence intervals. The delivery ratio with $p_{dard} = 0$ is at 51% much lower than the 77.6% delivery ratio reported at similar speeds during the backtracking experiments (section 5.4). The only difference between the two sets of experiments are the rate at which data packets are generated – packets in this section's network are generated 10 times faster than in the backtracking experiments. It appears that the higher network load due to faster packet generation is causing a substantial increase of 27.6% in the number of packets being dropped.

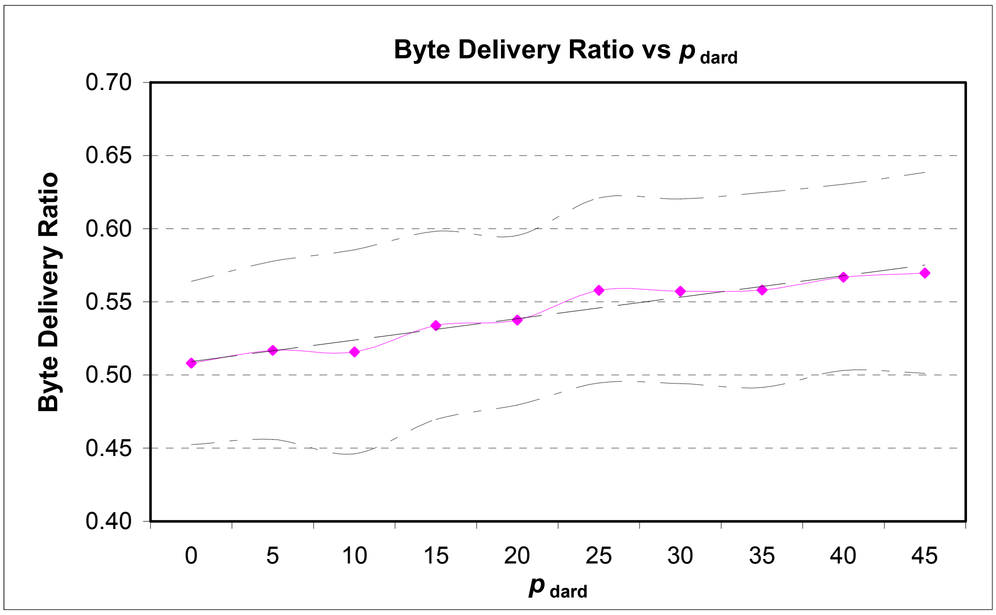


Figure 27 : Impact of p_{dard} on byte delivery ratio

Table 15 : Impact of p_{dard} on byte delivery ratio

p_{dard}	*Delivery Ratio*		σ
0	0.51	+- 0.056	0.090
5	0.52	+- 0.061	0.098
10	0.52	+- 0.070	0.113
15	0.53	+- 0.064	0.104
20	0.54	+- 0.058	0.094
25	0.56	+- 0.063	0.102
30	0.56	+- 0.063	0.102
35	0.56	+- 0.067	0.107
40	0.57	+- 0.064	0.103
45	0.57	+- 0.069	0.111

The increase in delivery ratio with higher p_{dard} values is disappointingly small. At $p_{dard} = 0.45$, the destination node will launch a RREQ packet roughly on every second data packet received. Yet there is only a 6% increase in byte delivery ratio. It is possible that the performance gains achieved by having more routes to the destination are offset by performance loss due to higher network load. It is also possible that the unicasting of a RREP packet is necessary to guarantee that the route consists of bidirectional links, and that the absence of RREP packets on routes discovered by the destination causes data packets to be dropped on low quality routes.

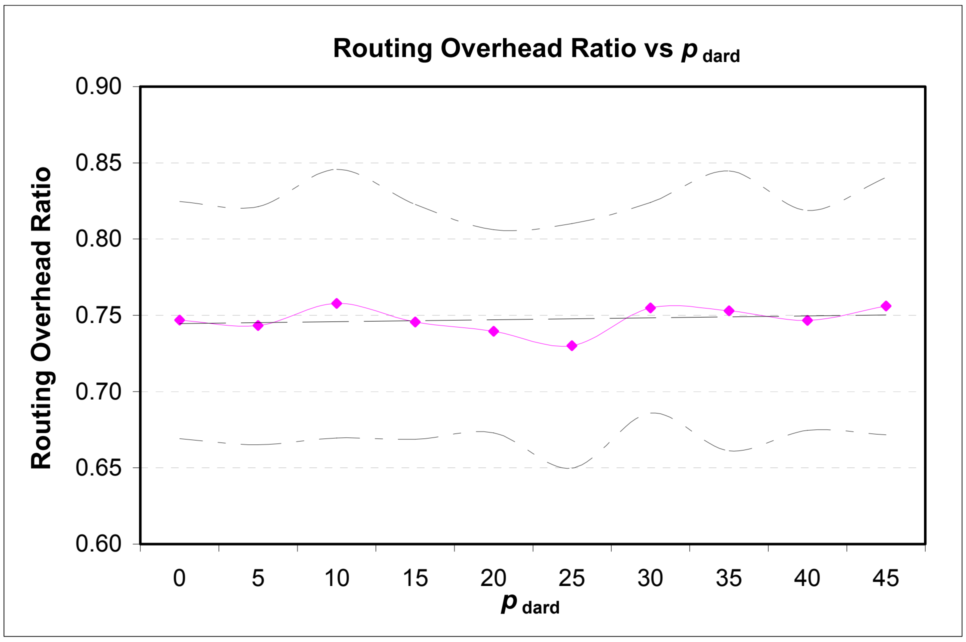
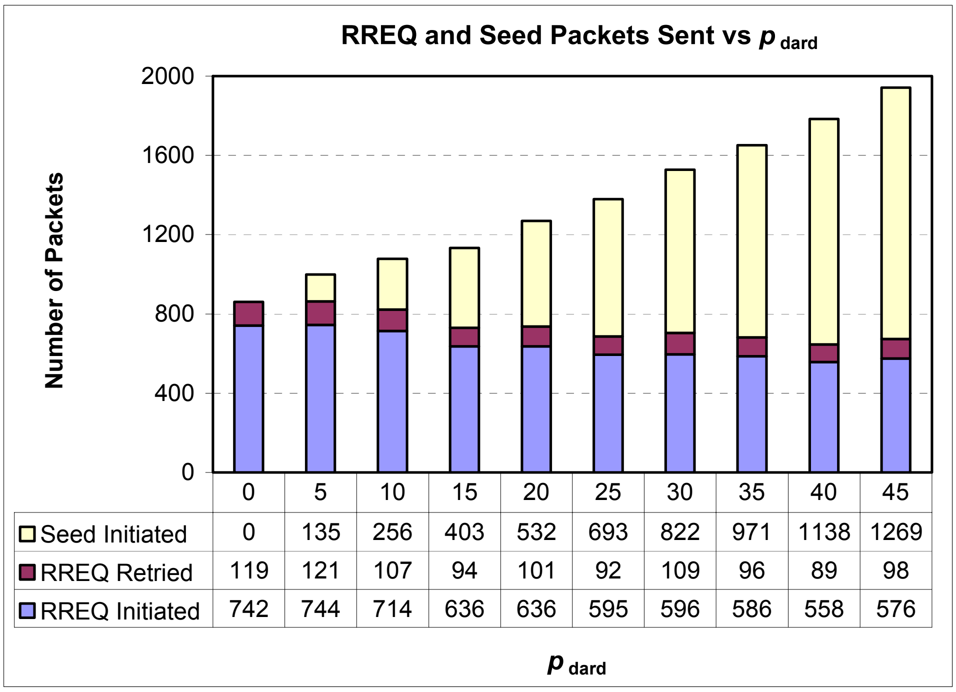


Figure 28 : Impact of p_{dard} on routing overhead ratio

Table 16 : Impact of p_{dard} on routing overhead ratio

p_{dard}	*Overhead Ratio*		σ
0	0.75	+- 0.078	0.125
5	0.74	+- 0.078	0.126
10	0.76	+- 0.088	0.142
15	0.75	+- 0.077	0.124
20	0.74	+- 0.067	0.107
25	0.73	+- 0.080	0.129
30	0.75	+- 0.069	0.111
35	0.75	+- 0.092	0.148
40	0.75	+- 0.072	0.116
45	0.76	+- 0.085	0.136

The routing overhead ratio depicted in Figure 28 is surprising. One would expect an increase in routing overhead ratio with increasing values of p_{dard}, as more RREQ packets are broadcast by destination nodes. Yet the routing overhead stays fairly constant for all values of p_{dard} - the maximum and minimum measured values only differ by 4%. Figure 29 shows the average number of RREQ packets initiated and retried as a function of p_{dard}. RREQ packets initiated by destination nodes are labeled as seed packets.



	0	5	10	15	20	25	30	35	40	45
□ Seed Initiated	0	135	256	403	532	693	822	971	1138	1269
■ RREQ Retried	119	121	107	94	101	92	109	96	89	98
■ RREQ Initiated	742	744	714	636	636	595	596	586	558	576

Figure 29 : Number of initiated RREQ packets

It could be argued that the RREQ packets broadcast by destination nodes simply substitute RREQ packets broadcast by source nodes, so that the resulting routing overhead ratio stays constant with changing values of p_{dard}. However, as can be seen in Figure 29 the average number of RREQ packets initiated and retried by source nodes only decrease slightly from 861 ($p_{dard} = 0$) to 674 ($p_{dard} = 0.45$). As p_{dard} is increased, the sum of RREQ and Seed packets sent increase from 861 to 1963, an increase of 128%. How is it possible that 128% more RREQ and Seed packets are being flooded, but the routing overhead ratio stays roughly constant?

It is possible that the extra routes continuously provided by destination nodes cause packet rebroadcast and backtracking to drop significantly. Additionally, since many nodes now have routes to the destination node, the number of hops traveled by RREQ and RREP packets in search of routes to the destination may be greatly reduced. Analysis of the contributing components of routing overhead (such as RREQ, RREP, RERR and backtracking) and of the number of hops traveled by RREQ and RREP packets should provide more understanding of this phenomenon.

The experiments were repeated to test the effects of DARD over a range of node mobility. Two sets of experiments were performed, with p_{dard} set to 0 in the first set and 0.25 in the second. In each case, node mobility was stepped from 0m/s to 100m/s in 10m/s increments, and experiments repeated 20 times with different random seed values. All other experimental parameters were the same as for the above experiments.

Figure 30 and Table 17 depicts the byte delivery ratios measured in the experiments with standard deviation and 95% confidence intervals. In the Route Reply Delivery Ratio experiments earlier in this section it was observed that packet delivery in a slightly mobile network is higher than in a stationary network. The same trend is evident in the byte delivery ratio, with peak delivery ratios in both sets of experiments occurring at 10m/s. Across the measured mobility range the protocol using DARD slightly outperforms the standard protocol, with an average delivery ratio increase of 8% and a maximum increase of 12.9% at 90m/s.

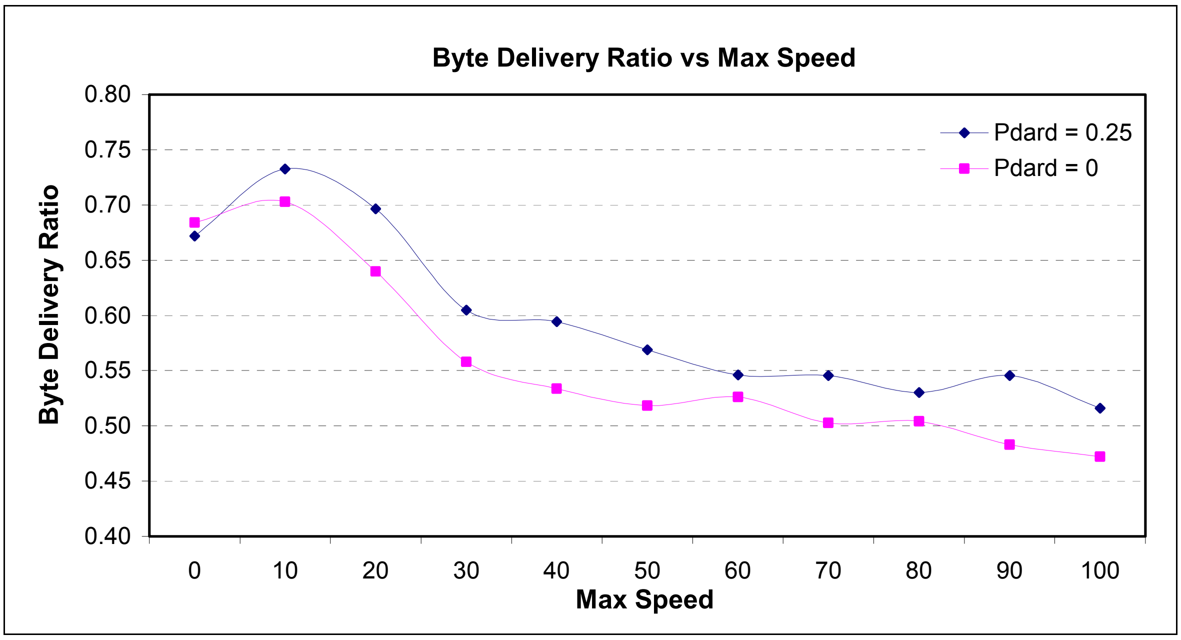


Figure 30 : Comparison of byte delivery ratio with/without DARD

Table 17: Comparison of byte delivery ratio with/without DARD

	$p_{dard} = 0$			$p_{dard} = 0.25$		
Max Speed	***Delivery Ratio***		***σ***	***Delivery Ratio***		***σ***
0	0.68	+- 0.089	0.144	0.67	+- 0.082	0.132
10	0.70	+- 0.090	0.145	0.73	+- 0.096	0.155
20	0.64	+- 0.107	0.172	0.70	+- 0.098	0.158
30	0.56	+- 0.114	0.183	0.60	+- 0.113	0.182
40	0.53	+- 0.090	0.146	0.59	+- 0.095	0.153
50	0.52	+- 0.065	0.106	0.57	+- 0.076	0.123
60	0.53	+- 0.058	0.094	0.55	+- 0.057	0.093
70	0.50	+- 0.048	0.077	0.55	+- 0.052	0.084
80	0.50	+- 0.041	0.066	0.53	+- 0.045	0.072
90	0.48	+- 0.045	0.073	0.55	+- 0.049	0.079
100	0.47	+- 0.040	0.065	0.52	+- 0.044	0.071

Figure 31 and Table 18 show the measured routing overhead ratios with standard deviation and 95% confidence intervals. At low mobility, link failures do not occur frequently and the extra routes provided by DARD is mostly unnecessary. Consequently the routing overhead of the protocol with DARD is higher than that of the standard protocol at low mobility. As node mobility increases the two curves are virtually identical. This again confirms the earlier observation that the DARD mechanism does not increase a network's routing overhead ratio. DARD may only provide a minor increase in performance, but at least it does so free-of-charge!

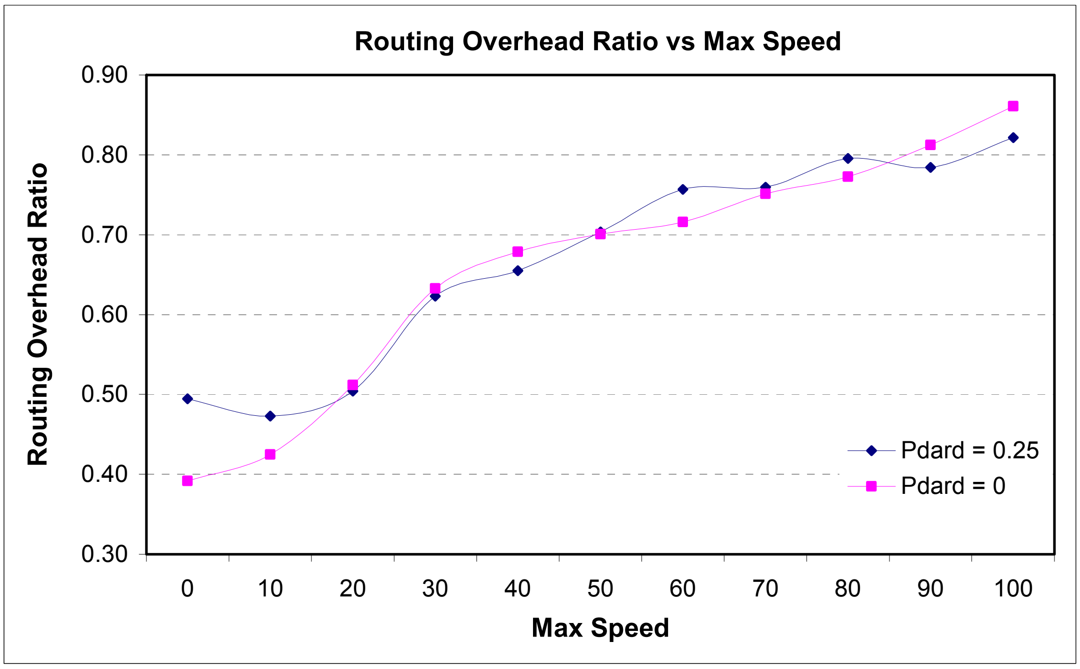


Figure 31 : Comparison of routing overhead ratio with/without DARD

Table 18 : Comparison of routing overhead ratio with/without DARD

	$p_{dard} = 0$			$p_{dard} = 0.25$		
Max Speed	***Overhead Ratio***		***σ***	***Overhead Ratio***		***σ***
0	0.39	+- 0.086	0.139	0.49	+- 0.070	0.113
10	0.42	+- 0.086	0.138	0.47	+- 0.089	0.143
20	0.51	+- 0.117	0.189	0.50	+- 0.091	0.146
30	0.63	+- 0.133	0.214	0.62	+- 0.109	0.177
40	0.68	+- 0.113	0.182	0.66	+- 0.100	0.161
50	0.70	+- 0.081	0.131	0.70	+- 0.084	0.136
60	0.72	+- 0.077	0.124	0.76	+- 0.064	0.104
70	0.75	+- 0.061	0.098	0.76	+- 0.067	0.108
80	0.77	+- 0.069	0.111	0.80	+- 0.071	0.114
90	0.81	+- 0.088	0.142	0.78	+- 0.067	0.109
100	0.86	+- 0.075	0.121	0.82	+- 0.067	0.107

The measured end-to-end delay curves are shown in Figure 32, with Table 19 indicating standard deviation and 95% confidence intervals. Across the measured mobility range DARD shows slightly lower end-to-end delay characteristics than the standard protocol. At high node mobility (60-100m/s) the end-to-end delay of the protocol using DARD only rises 10% from 0.20 to 0.22 seconds, while the end-to-end delay of the standard protocol rises by 48% from 0.25 to 0.37 seconds. It appears that fewer packets are delayed due to backtracking and that route replies are received faster. This observation strengthens the earlier notion that RREQ/RREP packets travel less hops when using DARD.

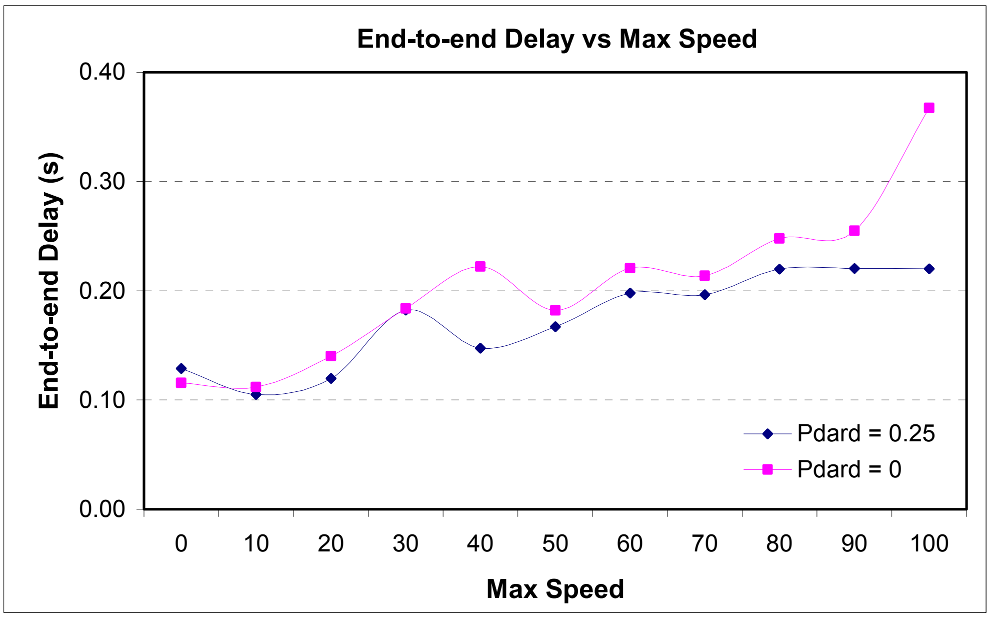


Figure 32 : Comparison of end-to-end delay with/without DARD

Table 19 : Comparison of end-to-end delay with/without DARD

	$p_{dard} = 0$			$p_{dard} = 0.25$		
Max Speed	***End-to-end Delay***		***σ***	***End-to-end Delay***		***σ***
0	0.12	+- 0.051	0.082	0.13	+- 0.042	0.067
10	0.11	+- 0.038	0.062	0.11	+- 0.024	0.038
20	0.14	+- 0.050	0.081	0.12	+- 0.031	0.050
30	0.18	+- 0.074	0.119	0.18	+- 0.072	0.117
40	0.22	+- 0.089	0.144	0.15	+- 0.036	0.059
50	0.18	+- 0.070	0.113	0.17	+- 0.057	0.092
60	0.22	+- 0.104	0.168	0.20	+- 0.060	0.097
70	0.21	+- 0.058	0.093	0.20	+- 0.062	0.099
80	0.25	+- 0.081	0.130	0.22	+- 0.065	0.105
90	0.25	+- 0.070	0.113	0.22	+- 0.088	0.142
100	0.37	+- 0.104	0.167	0.22	+- 0.075	0.121

Further study into the effects of DARD is certainly justified, including comparison of node power consumption characteristics and the number of routes available to destinations at each intermediate node. However, this section has shown that there are definite shortcomings in the traditional RREQ/RREP mechanism, and that an extension such as DARD provides marginally increased byte delivery ratio and end-to-end delay increases at no extra overhead cost.

5.7 Making next-hop decisions

Various previous works have shown that the use of ACO to make next-hop decisions offer improved performance compared to next-hop decisions based purely on hop-count. The aim of this section is to examine the influence of ACO parameters on a routing algorithm's performance.

The experiments performed in this section will start off with an algorithm where the probability of choosing a next-hop node is calculated using equation (5.4). This probability equation is similar to the equation used in the SACO algorithm [137], but without the pheromone amplification factor α. Once the probabilities are calculated, the next-hop node is selected using roulette-wheel selection.

$$p_{kn}^{d}(t) = \begin{cases} \dfrac{\tau_{kn}(t)}{\sum_{x \in N_k(t)} \tau_{kx}(t)} & \text{if } j \in N_k(t) \\ 0 & \text{if } j \notin N_k(t) \end{cases} \tag{5.4}$$

where:

t	=	time index.
$p_{kn}{}^{d}(t)$	=	the probability that a packet at node *k* at time *t* traveling to node *d* will be forwarded to next-hop node *n*, $0 < p \leq 1$.
$\tau_{kn}(t)$	=	the pheromone level at node *k* at time *t* associated with using node *n* as a next-hop node.
$N_k(t)$	=	the set of one-hop neighbours to node *k* at time *t* through which a route to the destination is known to exist.

A packet arriving at a node will increase the amount of pheromone at that node by an amount $\Delta\tau$ where:

$$\Delta\tau_{ki}^{s}(t) = \frac{k_l}{d_{ks}} \tag{5.5}$$

where:

$\Delta\tau_{ki}{}^{s}(t)$	=	the amount by which the pheromone associated with a transition from node *k* to node *i,* for a packet originating at node *s,* is incremented at time *t,* $\Delta\tau_{ki}{}^{s} \geq 0$.
d_{ks}	=	the distance in hops between node *s* and node *k*.
k_l	=	a tunable system parameter that controls the amount of pheromone deposited.

To simulate pheromone decay, pheromone values at each node should be continuously decreased as a function of time. However, since the nodes are discrete-time processes, pheromone levels can only be updated at discrete time intervals. Moreover, the computational expense incurred in decreasing the pheromone levels justifies not decreasing the pheromone levels as often as possible, but rather introducing a system parameter to regulate the interval at which this calculation should be performed. If this interval is set to a very short time, decreased pheromone levels will be calculated frequently. This will ensure that the values are always close to their ideal values, but at considerable computational expense, which also means higher power consumption. If this interval is set to a longer time, the pheromone levels will be calculated less frequently which could lead to “stale” or outdated pheromone levels being used in probability calculations, but with lower computational expense and lower power consumption.

Let ε denote the pheromone update interval. If the pheromone value at time t was calculated as $\tau(t)$, then the pheromone at time $t + \varepsilon \cdot \Delta t$ is calculated using:

$$\tau(t + \varepsilon \cdot \Delta t) = (1 - \rho)^{\varepsilon} \cdot \tau(t) \tag{5.6}$$

where:

ρ	= pheromone evaporation rate, $1 \geq \rho \geq 0$.

Note that pheromone levels increase linearly according to equation (5.5), but decrease exponentially according to equation (5.6).

To prevent unbounded accumulation of pheromone levels, a system-wide maximum pheromone parameter τ_{max} is introduced. Any pheromone level above this value is set equal to τ_{max}. In addition a system-wide minimum pheromone parameter τ_{min} is introduced. Any route on which the pheromone level decays to below this value is deleted. Note that this is similar to the Max-Min Ant System (MMAS) introduced by Stützle and Hoos [170] to address premature stagnation of the ACO algorithm.

5.7.1 Pheromone evaporation rate

Pheromone evaporation allows an ACO algorithm to "forget" old solutions gradually over time. In the case of ad hoc network routing algorithms pheromone evaporation plays a dual role:

- Pheromone evaporation allows routes to become less attractive over time so that "stale" routes are less likely to be used; and
- When the pheromone level on a route evaporates to a preset minimum value the route is deleted.

For small values of the pheromone evaporation constant ρ, pheromone evaporates slowly. Nodes will therefore accumulate more routes in their routing tables, but the routes may not be valid anymore. For large values of ρ the routes in a node's routing table are more likely to be valid, but the node may delete valid routes before they can be exploited. In this sense, there exists an analogy between pheromone evaporation and the route lifetime timers used by AODV.

The purpose of this section is to evaluate the influence of ρ on a protocol's performance. Although it is clear that optimum values of ρ will depend on node mobility, it is not clear what these optimum values are or how sensitive the algorithm is to sub-optimal values. To address these questions, experiments were performed where the value of ρ was stepped from 0 to 1 in 0.1 increments. The experiments were first done on a network with relatively high node mobility (60 m/s) and then repeated on a network with relatively low node mobility (10 m/s).

In each case, experiments were repeated 20 times with different random seed values. The pheromone update interval ε was set to 1 second in the experiments. In the experiments where $\rho = 1$, this implies that all nodes cleared their routing tables every second. If ε was set to a small enough value in the experiments with $\rho = 1$, no packets would be routed as routes would be deleted before they could be used. Also, very small settings of φ may not be achievable on small microcontrollers. The experimental parameters are summarised in Table 20.

Table 20: 100-node experimental setup with different ρ values

Parameter	***Run 1***	***Run 2***
CBR source packet generation rate	Random from 1 to 3 seconds	
Random Waypoint maximum speed (m/s)	60m/s	10m/s
ρ	Stepped from 0 to 1 in 0.1 steps	
ε	1 second	
General	Route reply by intermediate nodes enabled Backtracking enabled $p_{dard} = 0.25$	

Figure 33 and Table 21 depicts the measured byte delivery ratios with standard deviation and 95% confidence intervals. At low mobility, the protocol manages to deliver 69% of the packets with $\rho = 0$. Since link failures occur infrequently, the routes accumulated by nodes stay valid much longer, so the need to remove old routes with the evaporation mechanism is small. However at high mobility, invalid routes accumulated at nodes cause packets to be dropped much more frequently – only 50% of the packets were delivered at $\rho = 0$. Since no pheromone evaporation occurs at $\rho = 0$ packets are forwarded along stale routes, and when link failures occur the packets are backtracked along equally stale routes.

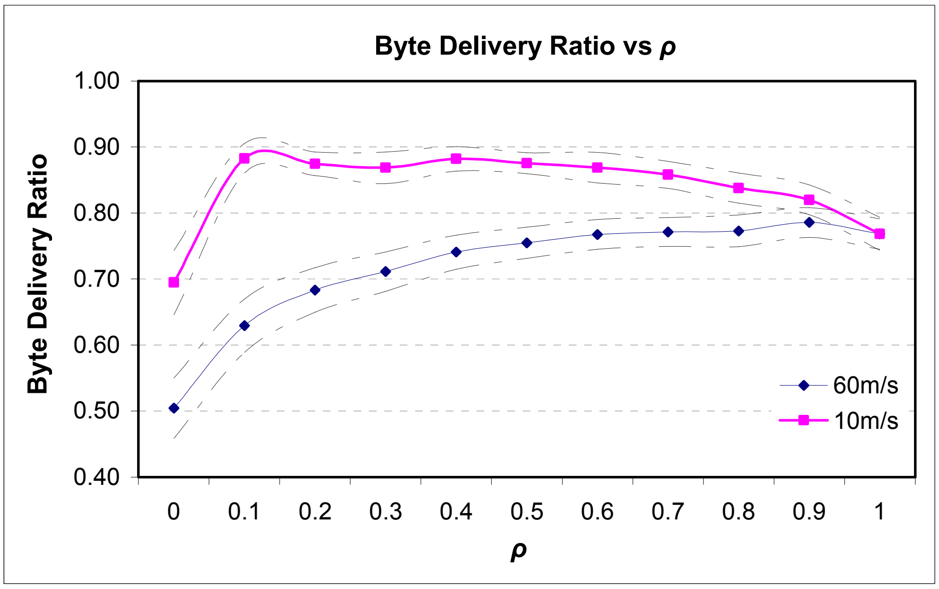


Figure 33 : Impact of ρ on byte delivery ratio

Table 21 : Impact of ρ on byte delivery ratio

	10 m/s			***60 m/s***		
ρ	***Delivery Ratio***		σ	***Delivery Ratio***		σ
0	0.69	+- 0.049	0.079	0.50	+- 0.046	0.074
0.1	0.88	+- 0.023	0.036	0.63	+- 0.041	0.065
0.2	0.87	+- 0.018	0.029	0.68	+- 0.034	0.054
0.3	0.87	+- 0.024	0.039	0.71	+- 0.030	0.048
0.4	0.88	+- 0.019	0.030	0.74	+- 0.026	0.042
0.5	0.88	+- 0.016	0.026	0.76	+- 0.024	0.038
0.6	0.87	+- 0.023	0.038	0.77	+- 0.022	0.036
0.7	0.86	+- 0.021	0.033	0.77	+- 0.022	0.035
0.8	0.84	+- 0.023	0.037	0.77	+- 0.024	0.039
0.9	0.82	+- 0.022	0.036	0.79	+- 0.023	0.036
1	0.77	+- 0.025	0.040	0.77	+- 0.023	0.037

At high mobility, higher values of ρ resulted in higher byte delivery ratios, since nodes are always using fresh routes. However at low mobility, the byte delivery ratio peaked where $0.5 \geq \rho \geq 0.2$. When $\rho > 0.5$ the byte delivery ratio started dropping as valid routes are being deleted before they can be used. At $\rho = 1$ the byte delivery ratio is similar at both high and low mobility. Since all routes are deleted every second, nodes always have to send a RREQ packet and immediately use the route when a RREP is received. This seems to indicate that, at high mobility, not many nodes moved out of each other's range in the one second pheromone update interval. It is expected that the delivery ratio with $\rho = 1$ will become lower as maximum mobility is increased beyond 60m/s.

The routing overhead ratios with standard deviation and 95% confidence levels are shown in Figure 34 and Table 22. At high mobility the routing overhead ratio is elevated at $\rho = 0$, and declines to a minimum at $\rho = 0.6$. This could be due to many packets being retried or backtracked on stale routes. This notion is also supported by the number of RREQ packets sent and retried (shown for high mobility in Figure 35). The number of RREQ packets sent rises very slightly from $\rho = 0$ to $\rho = 0.5$, but the routing overhead ratio decreases sharply in the same interval, indicating that fewer packets are retried and backtracked. At low mobility, routes last much longer and any route that gets removed through pheromone decay may well be a valid route. Consequently, routing overhead rises continuously with higher values of ρ.

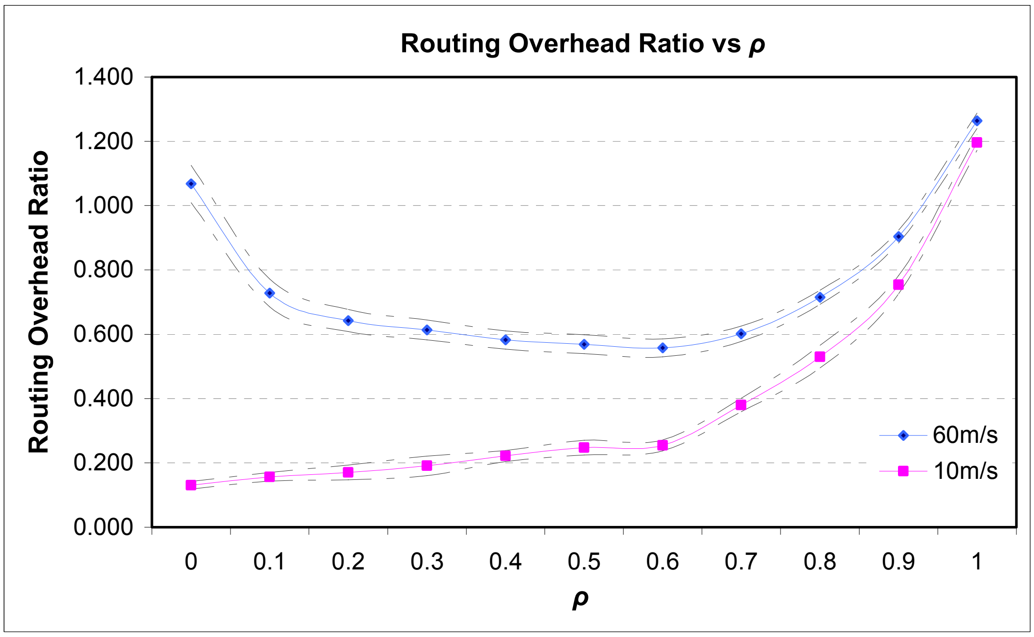
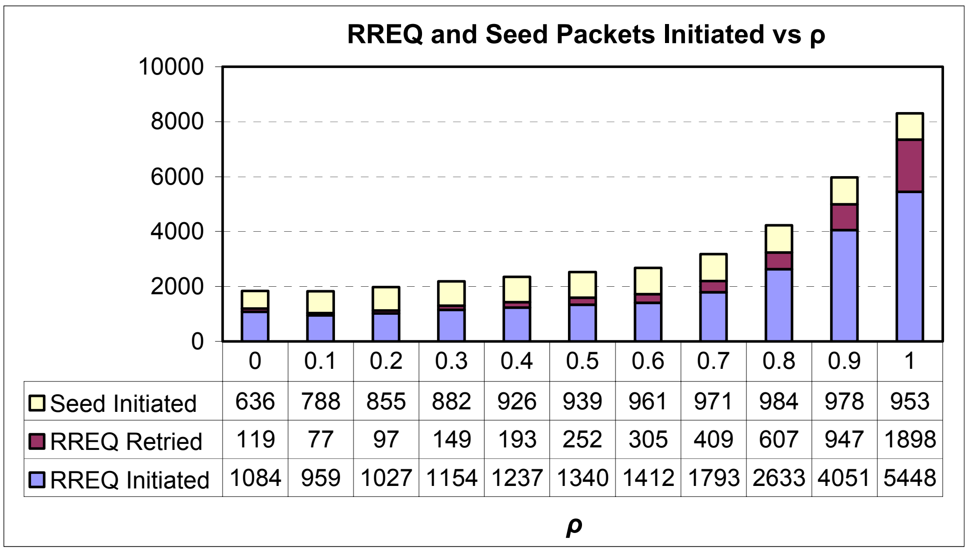


Figure 34 : Impact of ρ on routing overhead ratio

Table 22 : Impact of ρ on routing overhead ratio

	10 m/s			*60 m/s*		
ρ	***Overhead Ratio***		***σ***	***Overhead Ratio***		***σ***
0	0.130	+- 0.012	0.019	1.068	+- 0.058	0.094
0.1	0.156	+- 0.014	0.022	0.728	+- 0.043	0.070
0.2	0.170	+- 0.023	0.038	0.643	+- 0.034	0.055
0.3	0.191	+- 0.030	0.049	0.614	+- 0.031	0.050
0.4	0.222	+- 0.018	0.028	0.582	+- 0.028	0.045
0.5	0.248	+- 0.023	0.037	0.569	+- 0.030	0.048
0.6	0.254	+- 0.017	0.027	0.558	+- 0.028	0.045
0.7	0.380	+- 0.021	0.034	0.601	+- 0.024	0.038
0.8	0.530	+- 0.035	0.057	0.715	+- 0.022	0.035
0.9	0.754	+- 0.028	0.046	0.904	+- 0.020	0.032
1	1.197	+- 0.025	0.040	1.264	+- 0.024	0.038

As can be expected, the routing overhead at high mobility is generally higher than at low mobility. However the two overhead ratios converge as $\rho \rightarrow 1$, as the protocol in both scenarios is forced to send a RREP each time a packet is to be sent. The overhead ratio in both cases stays relatively stable in the interval $0.5 \geq \rho \geq 0.2$. The two curves show minima of 0.56 at $\rho = 0.6$ for the high mobility scenario, and 0.13 at $\rho = 0$ for the low mobility scenario. In both cases, the overhead ratio starts rising sharply for $\rho > 0.7$ as more and more valid routes are deleted.



	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
🟨 Seed Initiated	636	788	855	882	926	939	961	971	984	978	953
🟥 RREQ Retried	119	77	97	149	193	252	305	409	607	947	1898
🟦 RREQ Initiated	1084	959	1027	1154	1237	1340	1412	1793	2633	4051	5448

Figure 35 : Number of initiated RREQ packets at 60m/s

The measured end-to-end delay values are shown in Figure 36 with standard deviation and 95% confidence intervals in Table 23. It is interesting that the end-to-end delay curve at high mobility does not have the same elevated levels at $\rho = 0$ as the routing overhead ratio. If the overhead ratio at $\rho = 0$ is raised due to more backtracking and retransmissions, then shouldn't the end-to-end delay also have the same characteristic? It should be kept in mind that only packets that successfully reach their destination nodes are included in the packet delay calculation (refer to the end-to-end delay definition in section 4.3.3). This seems to indicate that, at high mobility, most packets that get backtracked and retransmitted at $\rho = 0$ get dropped before they reach the destination, and therefore have no influence on the end-to-end delay. This is also supported by the sharp drop in high mobility byte-delivery ratio at $\rho = 0$ observed in Figure 33.

The end-to-end delay is generally slightly higher in the high mobility scenario, as link failures (and their associated delays) are more likely to occur. The delays in both cases stays relatively stable, only rising slightly in the interval $0.7 \geq \rho \geq 0$. When $\rho > 0.7$ the delays in both cases start rising sharply (similar to the sharp rise in routing overhead ratio) as increasing numbers of packets have to wait for a RREQ/RREP sequence before being sent.

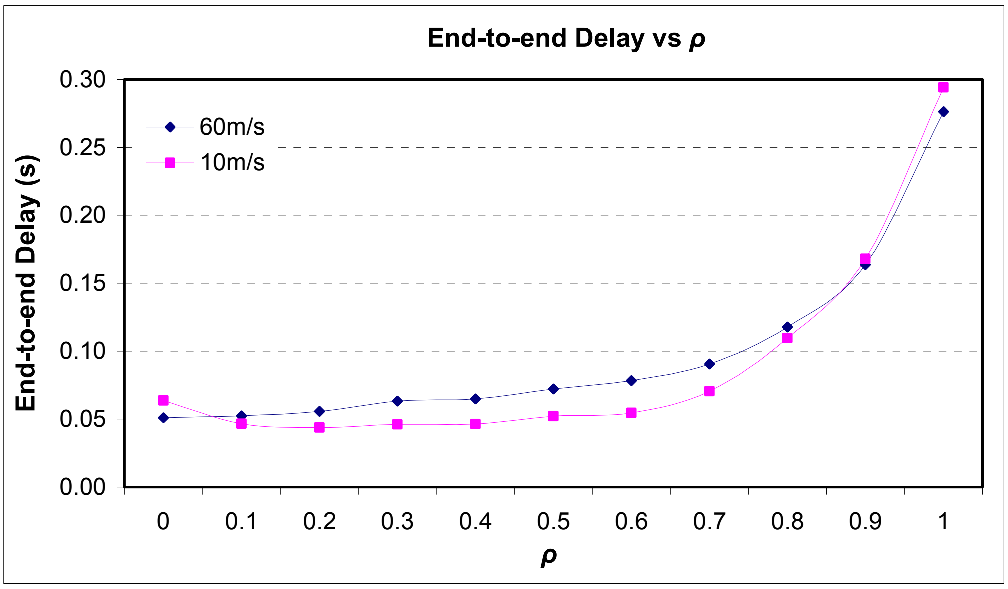


Figure 36 : Impact of ρ on end-to-end delay

Table 23 : Impact of ρ on end-to-end delay

	10 m/s			*60 m/s*		
ρ	*End-to-end Delay*		*σ*	*End-to-end Delay*		*σ*
0	0.064	+- 0.005	0.008	0.05	+- 0.006	0.009
0.1	0.047	+- 0.005	0.007	0.05	+- 0.006	0.009
0.2	0.044	+- 0.005	0.007	0.06	+- 0.005	0.009
0.3	0.046	+- 0.005	0.008	0.06	+- 0.007	0.011
0.4	0.046	+- 0.005	0.009	0.06	+- 0.006	0.010
0.5	0.052	+- 0.008	0.013	0.07	+- 0.007	0.012
0.6	0.055	+- 0.006	0.009	0.08	+- 0.009	0.014
0.7	0.071	+- 0.010	0.016	0.09	+- 0.008	0.013
0.8	0.110	+- 0.015	0.025	0.12	+- 0.011	0.018
0.9	0.168	+- 0.020	0.032	0.16	+- 0.010	0.016
1	0.294	+- 0.029	0.047	0.28	+- 0.022	0.035

All three measured metrics do not appear to be overly sensitive to slightly suboptimal values of ρ. In both the high and low mobility scenarios, any setting of ρ in the range $0.5 \geq \rho \geq 0.3$ yields more-or-less similar results.

The optimal setting of ρ is at 0.6 for the high mobility scenario, and at 0.1 for the low mobility scenario. However if ρ is set to 0.4 in both cases, all the metrics are still on average within 5% of their optimal values. If a mechanism is implemented to make nodes mobility-aware (such as counting the number of link failures), then the value of ρ can be dynamically adjusted by each node. However, in the observed mobility range such a mechanism would only provide marginally improved performance.

5.7.2 Pheromone amplification

When link failures occur, a routing protocol that employs ACO to make next-hop decisions suffer from problems similar to the *shortcut problem* and *blocking problem* described in section 2.2. In order to effectively manage link failures, a protocol must spend some effort on exploring new routes while exploiting (i.e. using) existing routes, so that backup routes are available when needed.

Because an ACO algorithm makes each next-hop decision probabilistically, some exploration occurs naturally. While packets will most probably exploit the route with the most pheromone, there remains a probability that the packets will explore routes with less pheromone. But how can the protocol's exploration/exploitation characteristic be explicitly controlled? In ACO algorithms used for problems such as the TSP [96], the algorithm's exploration/exploitation characteristic is partially controlled by the pheromone evaporation constant ρ. For small values of ρ pheromone evaporates slowly, consequently more pheromone accumulates on links, and the ants are more likely to follow existing routes. For large values of ρ pheromone evaporates faster, less pheromone accumulates on links, and the ants are more likely to explore other routes. In an ad hoc routing algorithm however, the pheromone tables represent information obtained at the cost of scarce battery and bandwidth resources. It would be advantageous if there was a way to control the algorithm's exploration without having to "forget" the information in the pheromone tables.

Equation (5.4), which is used to calculate next-hop probabilities, does not include the pheromone amplification constant α which is present in the original binary-bridge experiment by Pasteels *et al* [136]. This constant controls the bias towards routes with higher pheromone levels when doing probability calculations. The value of α influences the algorithm's exploration/exploitation characteristic, since higher levels of α will cause more ants to follow routes with high pheromone and fewer ants to explore other routes. When $\alpha = 0$, pheromone levels are ignored in the probability calculations and the ACO algorithm reduces to a random search.

The first ACO algorithm to use α was the SACO algorithm [137]. Equation (5.4) is now extended to resemble the probability calculation used by SACO, but with variable definitions adapted to an ad hoc network:

$$p_{kn}^{d}(t)=\begin{cases}\dfrac{\tau_{kn}^{\alpha}(t)}{\sum_{x\in N_k}\tau_{kx}^{\alpha}(t)} & \text{if } j\in N_i^k\\ 0 & \text{if } j\notin N_i^k\end{cases} \tag{5.7}$$

where:

t	=	time index
$p_{kn}{}^{d}(t)$	=	the probability that a packet at node *k* at time *t* traveling to node *d* will be forwarded to next-hop node *n*, $1\geq p_{kn}^{d}\geq 0$
α	=	a positive constant that amplifies the influence of pheromone concentrations.
$\tau_{kn}(t)$	=	the pheromone level at node *k* at time *t* associated with using node *n* as a next-hop destination.
$N_k(t)$	=	the set of one-hop neighbours to node *k* at time *t* through which a route to the destination is known to exist.

An alternative method of controlling exploration/exploitation is suggested by Schoonderwoerd *et al* [71], based on a *pseudo-random-proportional* action rule. This concept was first introduced by Gambardella and Dorigo [98] in Ant Colony System (ACS) to explicitly control an algorithm's exploration/exploitation characteristic. However, in ACS the rule is used to choose between a short link with high pheromone levels (chosen deterministically) and a random link using pheromone probabilities. Schoonderverwoerd *et al* suggests using this rule to choose between a purely random node and a node selected randomly using pheromone probabilities.

A protocol using this next-hop decision rule will therefore select a next-hop node using:

$$\text{Next-hop Node}=\begin{cases}\text{According to ACO probability } p_{kn}^{d} \text{ if } U(1)\geq p_{explore}\\ \text{Randomly from } N_i^k \text{ if } U(1)< p_{explore}\end{cases} \tag{5.8}$$

where:

$p_{explore}$	=	ACO algorithm exploration probability, $1\geq p_{explore}\geq 0$.
$N_k(t)$	=	the set of one-hop neighbours to node *k* at time *t*.

For small values of $p_{explore}$, the algorithm is likely to choose a next-hop node using the ACO probabilities. For larger values of $p_{explore}$, the probability of choosing a random next-hop node increases and the algorithm consequently explores more routes.

An important difference should be noted between the above two mechanisms. The first mechanism chooses routes only from the set of nodes through which a route to the destination is known to exist. The second mechanism chooses a next-hop node from the set of one hop neighbours (with the no-return rule). The second mechanism can therefore forward a packet to a node which is not known to have any routes to the destination. The second mechanism will explore more aggressively, but it may forward packets to invalid nodes.

This section will only investigate the effect of different values of α on a routing protocol's performance, and will not consider the second mechanism. A set of experiments were performed to evaluate the impact of the pheromone amplification constant α on a protocol's performance. In the experiments, maximum node mobility was first set to a relatively high level of 60m/s, and the value of α stepped from 0 to 2 in steps of 0.2. The experiments were then repeated with relatively low node mobility of 10m/s. Table 24 summarises the experimental parameters.

Table 24: 100-node experimental setup with different α values

Parameter	***Run 1***	***Run 2***
CBR source packet generation rate	Random from 1 to 3 seconds	
Random Waypoint maximum speed (m/s)	60m/s	10m/s
α	Stepped from 0 to 2 in 0.2 steps	
General	Route reply by intermediate nodes enabled Backtracking enabled $p_{dard} = 0.25$ $\rho = 0.4$ ε = 1 second	

The byte delivery ratios observed in the two sets of experiments are shown with standard deviation and 95% confidence intervals in Figure 37 and Table 25. In both cases, there do not appear to be a clear relationship between α and the delivery ratio.

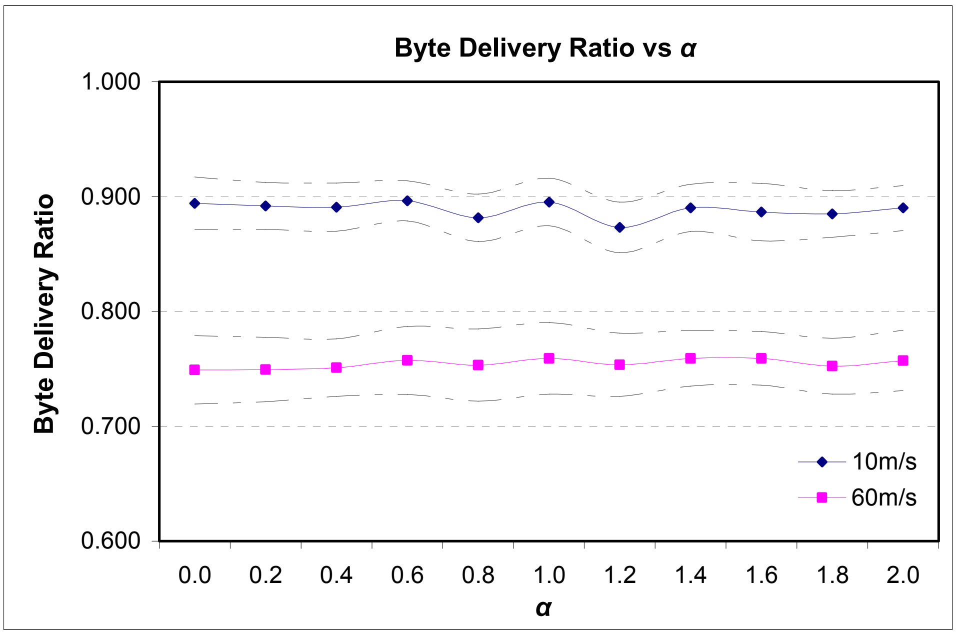


Figure 37 : Impact of *α* on byte delivery ratio

Table 25 : Impact of *α* on byte delivery ratio

	10 m/s			***60 m/s***		
α	***Delivery Ratio***		***σ***	***Delivery Ratio***		***σ***
0.0	0.894	+- 0.023	0.037	0.749	+- 0.030	0.048
0.2	0.892	+- 0.020	0.033	0.749	+- 0.028	0.045
0.4	0.891	+- 0.021	0.034	0.751	+- 0.025	0.040
0.6	0.896	+- 0.018	0.028	0.757	+- 0.030	0.048
0.8	0.881	+- 0.021	0.033	0.753	+- 0.031	0.051
1.0	0.895	+- 0.021	0.033	0.759	+- 0.031	0.050
1.2	0.873	+- 0.022	0.036	0.754	+- 0.028	0.044
1.4	0.890	+- 0.021	0.033	0.759	+- 0.024	0.039
1.6	0.887	+- 0.025	0.040	0.759	+- 0.023	0.038
1.8	0.885	+- 0.020	0.033	0.752	+- 0.024	0.039
2.0	0.890	+- 0.020	0.032	0.757	+- 0.026	0.042

The routing overhead ratios observed in the two sets of experiments are shown in Figure 38, with standard deviation and confidence levels indicated in Table 26. At high node mobility, the overhead appears to increase slightly with higher values of *α*, then decrease slightly for $\alpha > 1.8$. At low node mobility there is no distinct relationship between the overhead ratio and the value of *α*.

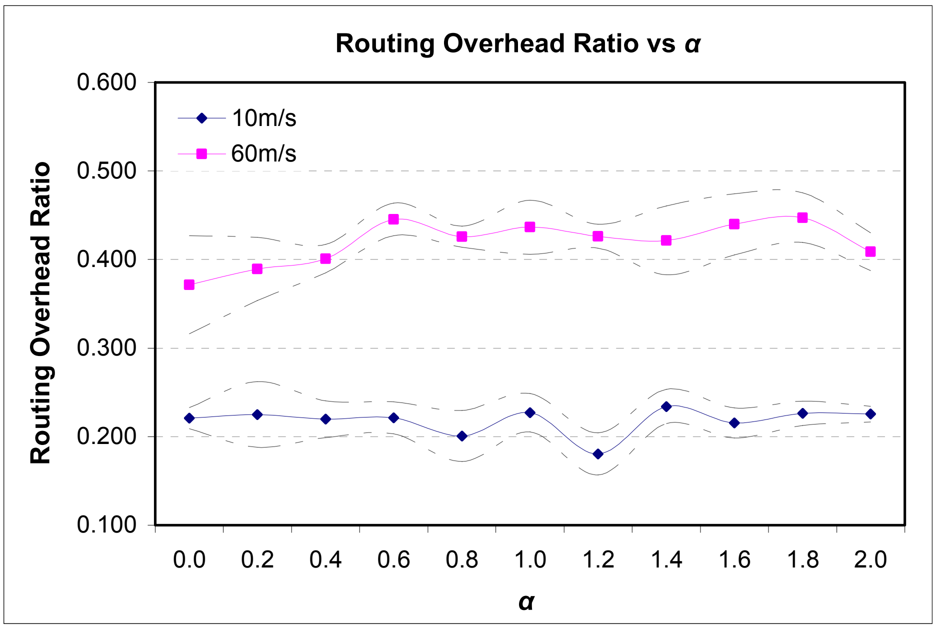


Figure 38 : Impact of *α* on routing overhead ratio

Table 26 : Impact of *α* on routing overhead ratio

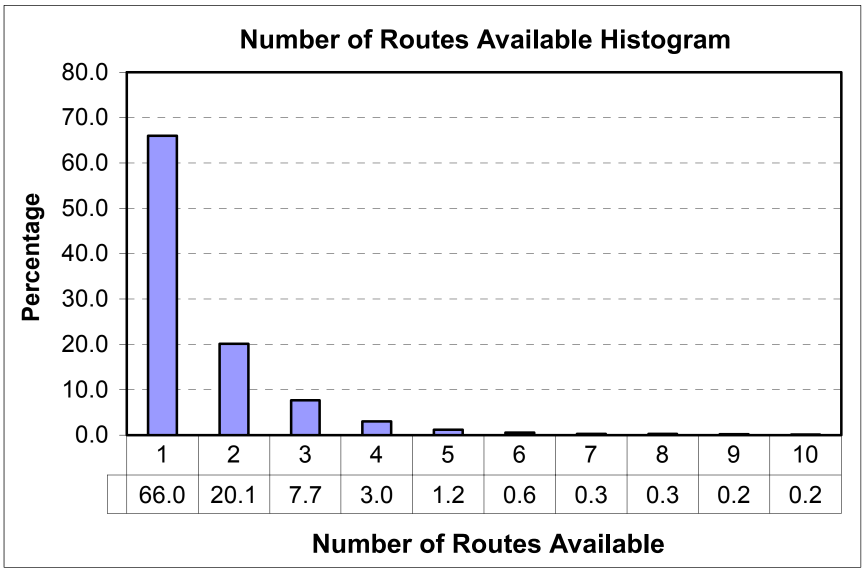
	10 m/s			*60 m/s*		
α	*Overhead Ratio*		*σ*	*Overhead Ratio*		*σ*
0.0	0.221	+- 0.012	0.019	0.371	+- 0.055	0.089
0.2	0.225	+- 0.037	0.060	0.389	+- 0.036	0.058
0.4	0.220	+- 0.021	0.034	0.401	+- 0.016	0.026
0.6	0.221	+- 0.018	0.029	0.445	+- 0.018	0.030
0.8	0.201	+- 0.029	0.047	0.426	+- 0.012	0.019
1.0	0.227	+- 0.022	0.035	0.436	+- 0.031	0.049
1.2	0.180	+- 0.024	0.038	0.426	+- 0.014	0.022
1.4	0.234	+- 0.019	0.031	0.422	+- 0.039	0.063
1.6	0.215	+- 0.017	0.028	0.440	+- 0.034	0.056
1.8	0.226	+- 0.014	0.022	0.447	+- 0.028	0.045
2.0	0.225	+- 0.009	0.014	0.409	+- 0.021	0.034

These results are puzzling to say the least. For low values of *α*, next-hop decisions are taken almost randomly from the set of neighbour nodes through which a route to the destination is known to exist. For high values, the link with the most pheromone will likely be selected. How can these two extremes yield such similar results?

To shed more light on these results, the number of routes available each time that a node makes a next-hop decision was recorded in a histogram to indicate the percentage of times that a specific number of routes were available. The experiment was repeated 20 times with *α* = 1 at relatively high node mobility of 60m/s, and the results averaged. The resulting histogram is shown in Figure 39.

When a next-hop decision needs to be made, the ACO algorithm chooses from a list of valid nodes. In 66% of the cases, this list contained only *one* entry, and in these cases the mechanism used to choose between options is meaningless since there is only one option to choose from. A further 20.1% of next-hop decisions were made with only 2 options available.

The current RREQ/RREP mechanism does not guarantee that routes are node-disjoint or link-disjoint. It is therefore possible that a node will have the choice between two routes that are virtually equivalent. When more routes are available the probability of the routes being more diverse increase. However, the algorithm only had 3 or more choices in 13.9% of the cases where next-hop decisions had to be made.



1	2	3	4	5	6	7	8	9	10
66.0	20.1	7.7	3.0	1.2	0.6	0.3	0.3	0.2	0.2

Figure 39 : Number of routes available histogram

These results indicate very strongly that the stochastic properties of an ACO algorithm has virtually no effect or advantage on the performance of the current routing algorithm, as most of the time there are just not enough options available. This is despite broadcast jitter (section 5.5) introduced for better broadcasting of RREQ packets, and the destination assisted route discovery mechanism (section 5.6) to provide more routes to the destination.

Table 27: Packet propagation mechanisms

Packet Type	***Propagation Mechanism***
RREQ	Flooded by broadcasting
RREP	Unicast stochastically using ACO algorithm
RERR	Unicast deterministically to upstream node
Data packet	Unicast stochastically using ACO algorithm
Backtracked data packet	Unicast deterministically along shortest route to source

The mechanisms by which packets are propagated through the network is summarised in Table 27. Data packets are unicast stochastically to their destination using the ACO algorithm. In addition, RREP packets are also unicast stochastically to the source using the ACO algorithm. However, for both these packets the ACO algorithm is mostly meaningless since there is frequently only one route available to destination or source.

In order for the ACO algorithm to provide significant results, the mechanisms used to provide routes have to be improved so that more routes are available. This can be done in the following ways:

- By sending more RREQ packets. Flooding the network with more RREQ packets may create more routes to the source at intermediate nodes, and destination nodes will respond with more RREP packets creating more routes to the destination. However, future work must evaluate the impact of this strategy on routing overhead ratio.
- By choosing next-hop nodes from all neighbour nodes (using the no-return rule), not only from the list of nodes through which a route to the destination is known to exist. This strategy will improve the protocol's exploration characteristic, but many packets may be dropped if they are sent to nodes that do not have a route to the destination.
- By using ant agents. Various works [85, 150, 171] have suggested this approach, where exploration agents separate from data packets are used to find new routes. The ant agents may create more routing overhead, but routing overhead may also be reduced by virtue of more routes being available. However, the seed packets generated by DARD provide a very similar service, and the resulting performance improvement has been shown to be marginal.
- By extracting more routing information from existing packets. The AOMDV algorithm [53] processes all received RREQ packets with the same sequence number, since RREQ packets may arrive at an intermediate node via different routes. This provides nodes with additional routes without adding overhead. It is also possible to activate *promiscuous listening* so that nodes may overhear packets not addressed to them, providing even more routing information at no extra overhead cost. However, both these mechanisms may cause transmission loops. The AOMDV algorithm therefore uses update rules based on the packet's sequence number and hop count to eliminate such loops.

5.8 Conclusion

This chapter tested various mechanisms that are frequently used by ad hoc routing algorithms. The chapter started off by clarifying the routing overhead ratio metric, so that meaningful comparison of the following sections was possible. The backtracking mechanism was then evaluated, and the results indicated that backtracking provides increasing byte delivery ratio improvements with increasing node mobility. It was also found that backtracking introduces additional routing overhead which is relatively constant with node mobility, and end-to-end delay which increases with increased node mobility.

The RREQ mechanism was then evaluated, and it was established that more RREQ packets do not necessarily lead to more routes. Insufficient broadcast jitter was shown to cause ineffective coverage of the network by RREQ packets, while excessive jitter may cause routes to become invalid before they are used.

The RREP mechanism was then evaluated, and it was found that a surprisingly low fraction of initiated RREP packets made it back to the source nodes, causing long end-to-end packet delay and increased routing overhead due to reinitiating of route requests. A mechanism was then introduced which actively involves the destination node in the route finding process by having the destination node broadcast special RREQ or "seed" packets. The impact of the DARD mechanism was evaluated, which revealed that the mechanism provided only slightly improved byte delivery ratios. Surprisingly, the DARD mechanism did not significantly increase routing overhead ratio, even at high values of p_{dard}.

The mechanism used to make next-hop decisions was then examined. An ACO algorithm was used to make next-hop decisions, and it was found that the algorithm was not very sensitive to different values of the pheromone evaporation constant ρ. Optimum values for ρ at both high and low mobility were experimentally obtained, and it was shown that ρ could be set to an intermediate value which is valid for both high and low mobility without significant performance loss.

The ACO algorithm was then extended to include the pheromone amplification constant α, and the impact of different values of α experimentally evaluated. It was found that α had no significant influence on the performance of the routing algorithm. It was then established that the failure of the RREQ/RREP and DARD mechanisms to provide multiple routes to destination nodes caused nodes to frequently have only 1 route available to choose from, so that the ACO algorithm is virtually meaningless. Mechanisms were then suggested that could increase the number of routes to destinations available, which would lead to more effective use of the ACO algorithm to make next-hop decisions.

The next chapter will use the knowledge from this and previous chapters to define a new multipath ad hoc routing algorithm, namely the Ant Colony Optimisation Distance Vector (ACODV) algorithm. The algorithm will then be benchmarked in various scenarios against the standard AODV algorithm.

6 The ACODV Algorithm

This chapter uses the knowledge gained from previous chapters and experimental work to define a routing protocol for very low-power environments. The features and operation of the protocol is described, as well as features not supported by the protocol. In addition, the protocol is benchmarked against AODV under varying mobility, load and network size conditions.

6.1 Introduction

This chapter defines the Ant Colony Optimisation Distance Vector (ACODV) routing algorithm for mobile ad hoc networks. This algorithm caters specifically for very low-power networks (such as sensor networks) where nodes may be required to operate for years without battery replacement.

Features of the ACODV algorithm are:

- ACODV is based on the AODV protocol, using the same distance vector and sequence number mechanisms as AODV.
- ACODV is a purely reactive protocol. The protocol does not use any *hello* messages or any other periodic messages, nodes only respond when they receive packets to process.
- ACODV is tailored to provide reliable communication, i.e. high byte delivery ratios. Where necessary, this is done at the cost of higher end-to-end packet delays and/or higher routing overhead.
- ACODV is a multipath protocol, but it does not offer any guarantee of node-disjoint or link-disjoint routes.
- ACODV uses pheromone levels to maintain routes, as opposed to route timers used by AODV.
- ACODV uses an ACO algorithm similar to SACO to forward packets.

The previous chapter investigated the performance of some routing sub-mechanisms. The results of these investigations are incorporated into ACODV in the following way:

- Backtracking of data packets is supported and is enabled by default, but may be disabled.
- The DARD mechanism is supported and is enabled by default, but may be disabled.
- Next-hop decisions are made using an ACO algorithm, but this feature can be disabled

individually or collectively on nodes so that next-hop decisions are made on lowest hopcount basis.

Disabling of the ACO algorithm on certain nodes means that the routing protocol can be implemented on nodes which do not have sufficient processing power to perform the necessary ACO calculations. Additionally, since the ACO algorithm is processing intensive a node using the ACO algorithm will consume more battery power than a node which uses lowest hopcount to make next-hop decisions. A node may therefore choose to switch to lowest hopcount decisions if the node detects that its battery is running low, without seriously affecting its neighbour nodes.

The rest of this chapter is organised as follows. Sections 6.2.1 and 6.2.2 describe the storing of routing information and the configuration parameters of ACODV, while section 6.2.3 describes the use and maintenance of sequence numbers. The packet format, generation and processing of RREQ, RREP, RERR and DERR packets are described in sections 6.2.4, 6.2.5, 6.2.6, and 6.2.7 respectively, with section 6.2.8 describing the processing and forwarding of data packets. The ACODV protocol's response to varying node mobility and network load is evaluated in sections 6.3.1 and 6.3.2, with section 6.3.3 evaluating the scalability of ACODV. Section 6.4 lists functions not supported by ACODV, and section 6.5 concludes this chapter.

6.2 Operation of ACODV

This section describes the general operation of the ACODV algorithm. The routing table, configuration parameters and sequence numbers used in ACODV are described in sections 6.2.1, 6.2.2 and 6.2.3 respectively. The control packets used by the ACODV algorithm are shown in Table 28, and are presented in more detail in sections 6.2.4, 6.2.5, 6.2.6 and 6.2.7. Finally, section 6.2.8 deals with the processing and forwarding of data packets.

Table 28: Control packets used by ACODV

Acronym	***Description***
RREQ	Route Request packet, broadcast in search of routes to a given destination node.
RREP	Route Reply packet, unicast in reply to a RREQ packet.
RERR	Route Error packet, unicast upstream on link failure.
DERR	A backtracked data packet, referred to as a *data error* packet.

6.2.1 Storing of Routing Information

The ACODV algorithm stores routing information in a routing table which is similar to the routing table used by the AOMDV algorithm [53], as shown in Table 29. For each known destination node, a list of potential routes to the destination is kept. Each potential route contains the following four elements:

$$(nexthop, hopcount, \tau, precursorlist) \tag{5.9}$$

where:

nexthop	=	the next-hop node to which packets on this route is forwarded.
hopcount	=	the hopcount to the destination using this next-hop node.
τ	=	the amount of pheromone accumulated on this route, $\tau_{max} \geq \tau \geq \tau_{min}$
precursorlist	=	a list of nodes from which packets that have used this route were received.

Table 29: ACODV routing table structure

Routing Table Entry	***Description***
Destination ID	A potential destination node's ID
Destination Sequence Number	The last-known sequence number of this destination
Route List	A list of routes to the destination in the form: $\{(nexthop_1, hopcount_1, \tau_1, precursorlist_1),$ $(nexthop_2, hopcount_2, \tau_2, precursorlist_2), \ldots\}$
DERR Sequence Number	The highest sequence number received in a DERR packet with this destination node as final destination.

In addition to the route list each entry in the routing table contains the last-known sequence number of the destination, used when forwarding data packets and the highest sequence number received in a DERR packet with this destination as final destination, used when forwarding DERR packets.

Consider a network with 1000 nodes where the maximum connectivity of each node (i.e. the maximum number of neighbour nodes) is not expected to exceed 10. A worst-case estimation of the routing table's memory requirement can be calculated as follows:

Assume that:

- Pheromone levels are stored as 2-byte values (yielding 65 535 distinct pheromone levels);
- Hopcounts are stored as 2-byte values, giving a maximum route length of 65 535 hops;
- Sequence numbers are stored as 2-byte values;
- IP addresses are 4 bytes long;
- Each route has 5 precursor nodes; and
- Each node has routes to 50% of the network (500 nodes) through 50% of its

neighbours (5 nodes).

Then the memory required by the routing table is calculated as:

$$500\left[(4+2+2)+5(2+2+5\times4)\right]=64000 \text{ bytes} \tag{5.10}$$

Roughly 64 kilobytes is a considerable amount of memory for most embedded applications. Some of the latest microcontrollers available at the time of writing, such as the AT91SAM7S64 from Atmel [172] and the LPC2106 from Phillips [173], have 64 kilobytes of RAM onboard the microcontroller, which would be insufficient for this scenario as there would be no memory left for program execution. However, the scenario above is seen as an extreme case, and it is believed that suitable microcontrollers can be found for most real-life applications.

In addition to the memory requirement, the routing table consumes considerable processing resources. The pheromone levels in the table have to be constantly maintained to simulate evaporation. This involves performing a calculation (see equation (5.6)) on each entry in the table. The pheromone update interval ε in equation (5.6) therefore plays an important role in tuning the algorithm's computational requirement to suit the node hardware.

6.2.2 Configuration Parameters

Table 30 lists all configuration parameters used by the ACODV algorithm.

Table 30: ACODV Configuration Parameters

Parameter Name	***Description***
Network Related	
network_diameter	The maximum route length (in hops) expected between any two nodes.
ttl_start	The initial TTL utilised by the expanding ring search.
ttl_increment	The amount by which expanding ring search TTL is incremented on each successive search.
ttl_end	The final TTL utilised by the expanding ring search.
Pheromone Related	
ε	The pheromone update interval as defined in equation (5.6).
ρ	The pheromone evaporation rate as defined in equation (5.6).
α	The pheromone amplification factor as defined in equation (5.7).
k_l	A constant that determines the amount of pheromone deposited as a function of the hopcount, defined in equation (5.5).
τ_{max}	The maximum level of pheromone that can accumulate in any entry in the routing table.
τ_{min}	The minimum level of pheromone on any entry in the routing table before the entry is deleted.
General	

Pdard	The probability that a node will launch RREQ packets on receipt of a data packet as final destination.

6.2.3 Maintaining Sequence Numbers

To ensure that ACODV is free of transmission loops, every node must maintain a list of sequence numbers. The sequence number of a node is incremented immediately before originating any packet and the incremented sequence number is added to the packet. The sequence number and the node ID form a unique combination to identify each packet.

When a node receives a packet, the node first checks the destination node information structure to see if this packet's sequence number is higher than any previously received packet from the originating node. If the sequence number of the packet is higher, the packet is processed and the sequence number field in this node's destination node information structure is updated with the sequence number from the just-received packet. If the sequence number in destination node information is higher than the packet's sequence number, the packet is silently discarded.

The sequence number mechanism precedes all other routing mechanisms described in the next sections. If a reference is therefore made in the next sections to a received packet, then the reference is to a packet that has been received and the sequence number checked.

6.2.4 Route Request (RREQ) packet generation, format and forwarding

A route request is initiated when the node has to communicate with a destination node for which no route exists. The originating node initiates the route request by broadcasting a RREQ packet to its neighbours. Table 31 lists the fields and format of a RREQ packet. Each neighbour node that receives the packet will respond with a RREP packet if the neighbour node has a valid route to the destination. If the neighbour does not have a valid route to the destination, the neighbour will increase the hop-count of the packet, decrease the packet's Time-to-live (TTL) by one, and rebroadcast the packet if the TTL is larger than zero. If the TTL is zero, the packet is deleted.

On sending the RREQ packet, the originating node sets a timer. If no response is received in RREQ_TIMEOUT milliseconds, the node re-initiates the route request process up to RREQ_RETRIES times. The packet is dropped if no response is received after all the retries.

The ACODV algorithm utilizes an *expanding ring* search by default. It has been shown [174] that such a search reduces network overhead without impacting byte delivery ratio negatively. However, an expanding ring search causes longer end-to-end packet delay times. For instances where such a search is not desired, the expanding ring search can be disabled by setting the *ttl_start* parameter equal to the network diameter.

Table 31: ACODV RREQ packet format

NrOfBytes	*Symbol*	*Description*
1	PktType	The packet type, in this case a RREQ packet
4	SrcID	Source node ID number
4	DestID	Destination node ID number
4	PrevID	The ID number of the node from which this packet was received
1	Flags	This field contains the *ReqRoutingAssistance* and *SendRREP* flags as described in section 5.6
2	HopCnt	The number of hops traveled by the packet from its originating node
4	SeqNr	The sequence number of this packet as assigned by the originating node
2	TTL	The number of hops remaining before this packet will be deleted

6.2.5 Route Reply (RREP) packet generation, format and forwarding

If a node receives a RREQ packet to a destination for which the current node does not have a valid route, the node rebroadcasts the RREQ packet as described in section 6.2.4. If the current node does have a valid route to the destination, or if the current node is the destination, then the node responds by sending a route reply (RREP) packet back to the originating node. Table 32 lists the fields and format of a RREP packet.

Two RREP cases can occur:

- **RREP by the destination node:** If the RREP is generated by the destination node itself, then the node copies its own ID number into the *SrcID, PrevID* and *DestID* fields of the RREP packet. The node's current sequence number is first incremented and then copied to the *SeqNr* and *DestSeqNr* fields. The hop-count from the just-received RREQ packet is copied to the *HopCnt* field.
- **RREP by an intermediate node:** If the RREP is generated by an intermediate node, the current node copies the last-known sequence number of the destination node (from the routing table) to the *DestSeqNr* field. This field is used by the originator of the RREQ packet to determine the "freshness" of routes. The destination node's ID number is copied to the *DestID* field, and the current node's ID number is copied to the *SrcID* and *PrevID* fields. The node's current sequence number is first incremented and then copied to the *SeqNr* field. The intermediate node calculates the hop-count to the destination node by adding the hop-count in the just-received RREQ packet and the distance from the intermediate node to the destination (as given by the intermediate node's routing table) together. The sum of these two distances is copied to the *HopCnt* field of the RREP packet.

When an intermediate node receives a RREP packet, the node first increments the hopcount field in the packet. The intermediate node creates or updates a route to the destination only if the *DestSeqNr* field in the RREP packet is higher than the destination sequence number already in the node's routing table. If the destination sequence numbers in the RREP packet and the routing table are the same, the route is updated only if the hopcount in the RREP packet is lower than the hopcount in the node's routing table. To avoid multiple RREP packets being forwarded along the same route, the node only forwards the RREP packet if it created or updated a route to the destination. RREP packets are forwarded using the ACO algorithm if enabled, or the shortest route if ACO is disabled.

The node that originated the RREQ process could potentially receive multiple RREP packets. If multiple RREP packets are received through different neighbours, the originating node will enter all the potential routes into its routing table. If more than one RREP is received though the same neighbour, the originating node will use the *HopCnt* field in the RREP packets to determine which packets contain the shortest routes to the destination, and will only update the routing table if the *HopCnt* field in any successive RREP is lower than the hopcount already in the routing table.

Table 32: ACODV RREP packet format

NrOfBytes	*Symbol*	*Description*
1	PktType	The packet type, in this case a RREP packet.
4	SrcID	Source node ID number.
4	DestID	Destination node ID number.
4	PrevID	The ID number of the node from which this packet was received.
2	HopCnt	The number of hops from the source to destination nodes.
2	SeqNr	The sequence number of this packet as assigned by the originating node.
2	DestSeqNr	The sequence number (or last-known sequence number) of the destination node.

6.2.6 Route Error (RERR) packet generation, format and forwarding

Routes in an ad hoc network can become invalid at any time due to node mobility or node failure. This causes a problem similar to the *blocking problem* (see section 2.2) in real ants. However, real ants deposit the same kind of pheromone when leaving and returning to the nest, and ants that are on a blocked trail will take longer to return to the nest, allowing the pheromone on the blocked trail to evaporate faster than the pheromone on a valid trail. In the ACODV algorithm, ants leaving a source node deposit pheromone pointing to the source node, and ants arriving from a destination node deposit pheromone pointing to the destination node. If an intermediate node in the route moves or fails, causing the route to become invalid, then the source node will be unaware that the route is invalid and will keep using the invalid route. Pheromone evaporation is therefore not a viable mechanism for indicating invalid routes in the ACODV algorithm.

The only case where pheromone evaporation may be sufficient to indicate an invalid route is where nodes require and *end-to-end* acknowledgement for every packet, i.e. a response from the destination node, before sending the next packet. In this case, the long delay caused by an invalid route may allow the pheromone on that route to evaporate sufficiently for the route to become invalid, or for another route to become more attractive. However, this may cause unacceptable packet delays, as the packets will be sent on invalid routes until sufficient pheromone decay has occurred.

The ACODV algorithm therefore uses route error (RERR) packets to indicate invalid routes, similar to RERR packets in the AODV algorithm. A RERR packet is sent when a link failure causes a potential destination node to become unreachable. Note that this is different from AODV where *any* link failure results in a RERR packet being sent. Table 33 lists the fields and format of a RERR packet.

Table 33 : ACODV RERR packet format

NrOfBytes	***Symbol***	***Description***
1	PktType	The packet type, in this case a RERR packet.
4	SrcID	Source node ID number.
4	PrevID	The ID number of the node from which this packet was received.
4	SeqNr	The sequence number of this packet as assigned by the originating node.
2	NrOfUnreachID	The number of unreachable nodes listed in this RERR message.
4· NrOfUnreachID	UnreachIDList	A list of unreachable node IDs.

When a node encounters a broken link, the node makes a list of destinations that are no longer reachable as a result of the broken link. This is done by going through the next-hop neighbour's column in the routing table, and listing the next-hop neighbour along with all destination nodes that are reached *only* through this next-hop neighbour. The node then deletes all these routes and broadcasts a RERR packet containing the list of unreachable nodes.

Any node that receives a RERR packet checks to see if it has any routes to the indicated unreachable nodes that use the node from which the packet was received as a next-hop node. If the receiving node has any such routes, then these routes are deleted. However, the node that received the RERR packet may be aware of other routes to the unreachable nodes. After deleting these routes, the receiving node does therefore not simply forward the RERR packet, but first checks to see if it has alternative routes to any of the unreachable nodes. If the receiving node does have alternative routes to any of the unreachable nodes then, from the receiving node's perspective, these nodes are not unreachable. The receiving node therefore removes these nodes from the RERR packet, and then broadcasts the packet. Note that the receiving node does not generate a new RERR packet with a new source ID and sequence number, the receiving node simply modifies the RERR packet before re-broadcasting it.

If a node receives a RERR packet and finds that it doesn't have routes to any of the unreachable nodes, or that it has alternative routes to all the unreachable nodes in the RERR packet, then the receiving node will not re-broadcast the RERR packet. The RERR packet does therefore not have a time-to-live (TTL) restriction, it is re-broadcast as long as it contains useful information.

As an illustration of the forwarding of RERR packets, consider the 7-node network in Figure 40 when node F fails or moves away. Node D attempts to send a data packet to node F, but the attempt fails and node D generates a RERR packet. Assume that node D is not aware of the route to node G through nodes C and E. Node D consequently lists nodes F and G as unreachable, and broadcasts the RERR packet. Nodes A and C receive the RERR packet, and node A deletes the routes to nodes F and G from its routing table. Node C, however, is aware of the route to node G through node E. Node C deletes the routes through node D to nodes F and G from its routing table, but can still reach node G through node E. Consequently node C rebroadcasts the RERR packet with only node F listed as unreachable. If node E is not aware of node F, then node E will not rebroadcast the RERR packet since none of the nodes currently in node E's routing table became unreachable. If node E is aware of node F, and previously had routes to node F through nodes C and G, then node E will not rebroadcast the RERR packet since it still has a route to node F through node G in its routing table. If node E did not have a route to node F through node C but was not aware of the route to node F through node G, then node E will rebroadcast the RERR packet with node F listed as unreachable.

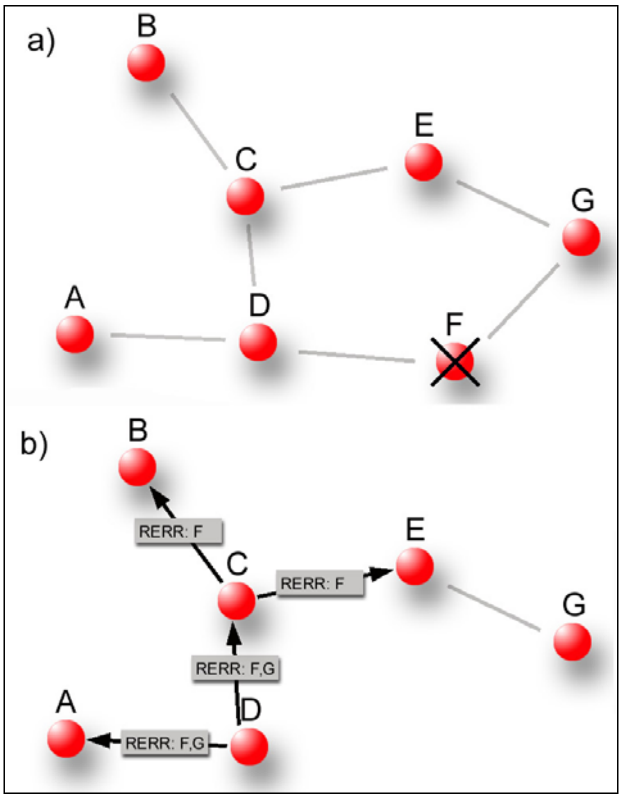


Figure 40 : Forwarding of RERR packets in ACODV

6.2.7 Data Error (DERR) packet generation, format and forwarding

A node generates a DERR packet when:

- A data packet is received from an upstream node but the node has no route to the destination, or
- All attempts to forward a data packet along routes in the routing table fails.

Note that the ACODV algorithm does not support *local repair* of routes yet, although this may be implemented in future work.

Table 34 lists the fields and format of a DERR packet. Before sending a DERR packet, the originating node copies the original packet into the DERR packet's *OriginalMsg* field. Note that the entire packet, including the original message destination and source IDs, are copied to the DERR packet so that upstream nodes may have all required information for further routing of the packet.

Table 34 : ACODV DERR packet format

NrOfBytes	***Symbol***	***Description***
1	PktType	The packet type, in this case a DERR packet.
4	PrevID	The ID number of the node from which this packet was received.
	OriginalMsg	The original data packet

A node originating a DERR packet forwards the DERR to the upstream node from which the original data packet was received. If the transmission fails, the DERR packet is forwarded along the shortest route known to exist to the source. If all attempts to backtrack the packet to an upstream node fail, the packet is dropped.

When a node receives a backtracked DERR packet, the node deletes all routes through the source of the DERR packet to the destination. The node then checks if the sequence number in the original packet contained in the DERR packet is higher than the sequence number for this destination recorded in the routing table. This is to prevent transmission loops of DERR packets. If the sequence number is invalid, the packet is dropped. If the sequence number is valid, the node checks if any alternative routes to the destination exist. If alternative routes exist, the packet is forwarded along one of the routes. If no alternative routes exist, the packet is further backtracked to an upstream node.

If the packet is backtracked to the original packet source, the source node may reinitiate a RREQ sequence for an alternative route to the destination or drop the packet and send an error message to the application layer.

6.2.8 Processing and Forwarding of data packets

When a source node receives a data packet from the application layer for processing, the node checks if route(s) to the destination exists in its routing table. If route(s) exist, the packet is forwarded using equation (5.7) if ACO is enabled, or along the shortest route if ACO is disabled.

If no route to the destination exists, the packet is buffered and a RREQ initiated. If no response is received after the RREQ process (optionally with expanding ring search) completes, the packet is returned to the application layer with an error message. When a node receives any packet with routing information (RREQ, RREP, RERR, DERR or data packet), the node checks if the packet contains a route required by any of its buffered packets. If a route required by a buffered packet is received, the packet is queued for transmission immediately.

When an intermediate node receives a data packet from an upstream node or a DERR packet from a downstream node, the node checks if route(s) to the destination exists in its routing table, and forwards the packet if valid route(s) exist. If no valid route to the destination exists, the intermediate node uses the backtrack mechanism described in section 6.2.7 to further process the packet.

6.3 Performance of ACODV

The previous sections defined the operation of ACODV. The aim of this section is to provide an overview of the performance of the ACODV protocol compared to AODV. The performance experiments performed in this section are by no means exhaustive and are only intended to give a brief overview of ACODV's performance. Future work will provide more in-depth performance studies and comparisons.

The rest of this section is organised as follows. Section 6.3.1 evaluates the response of ACODV and AODV to varying node mobility scenarios, while section 6.3.2 looks at the response of both protocols to varying network load. Lastly section 6.3.3 evaluates the scalability of both protocols to large networks.

6.3.1 Response to varying node mobility

The mobile nature of ad hoc networks adds a unique challenge to the routing function. Data packets must be delivered reliably to their destinations, but the routes required to deliver the packets are constantly changing. In this section it will be established whether ACODV manages to efficiently route packets in these dynamic environments.

As node mobility is simulated using the random waypoint model, the speed of a node at any given time is a random value between specified minimum and maximum speeds. In the experiments performed the minimum speed was set to 0 m/s, and the maximum speed stepped from 0 m/s (stationary nodes) to 100 m/s in 10 m/s increments. The mobility was not increased beyond 100 m/s as this corresponds to a maximum speed of 360 km/h, and it is believed that networks with nodes moving at over 360 km/h will have particular requirements not offered by ACODV. Experiments were performed for AODV and ACODV, and in each case the experiments were repeated 20 times with different random seed values and node starting positions. Table 35 presents an overview of the experimental parameters.

Table 35: Experimental setup for mobility experiments

Parameter	***Value***
Number of nodes in network	100
Node initial placement	Random
Number of CBR sources	5
CBR source start time	Random up to 10 seconds
Number of data packets created by each CBR source	1000
Data packet size	1024 bytes
Terrain size	3000m x 1000m
Random Waypoint pause time (s)	1
Random Waypoint maximum speed (m/s)	Stepped from 0m/s to 100m/s in 10m/s increments
Number of times experiment repeated	20
General	Route reply by intermediate nodes enabled Backtracking enabled $p_{dard} = 0.25$ $\rho = 0.4$ $\varepsilon = 1$ second

The observed byte delivery ratios with standard deviation and 95% confidence levels are shown in Figure 41 and Table 36. At very low mobility the delivery ratios of both algorithms are similar. As mobility was increased, ACODV consistently delivered from 5.5% to 10% more packets than AODV. Maintaining multiple routes and backtracking failed packets clearly contributes towards a more robust protocol, and makes ACODV deliver packets more reliably than AODV. What remains to be seen is at what cost this improvement is achieved.

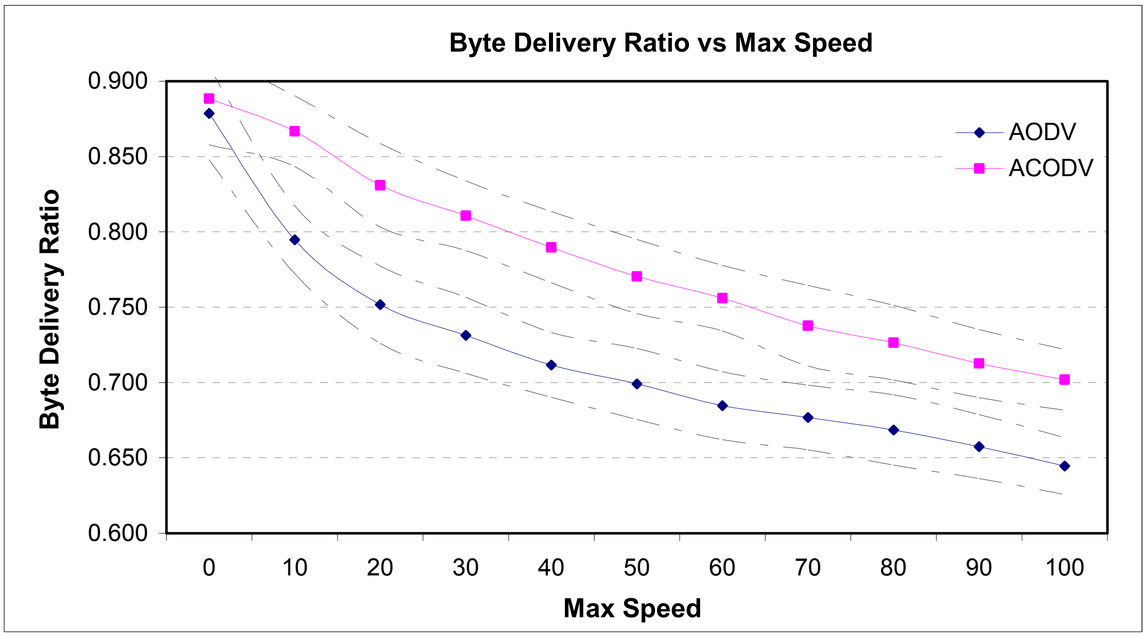


Figure 41 : Impact of mobility on byte delivery ratio

Table 36 : Impact of mobility on byte delivery ratio

	AODV			*ACODV*		
Max Speed	***Delivery Ratio***		***σ***	***Delivery Ratio***		***σ***
0	0.879	+- 0.031	0.049	0.888	0.031	0.049
10	0.795	+- 0.023	0.036	0.867	0.023	0.038
20	0.752	+- 0.026	0.041	0.831	0.028	0.045
30	0.731	+- 0.025	0.041	0.811	0.023	0.037
40	0.712	+- 0.022	0.035	0.790	0.024	0.038
50	0.699	+- 0.024	0.038	0.770	0.025	0.040
60	0.685	+- 0.023	0.036	0.756	0.022	0.035
70	0.677	+- 0.021	0.035	0.738	0.027	0.043
80	0.669	+- 0.023	0.038	0.726	0.025	0.040
90	0.658	+- 0.021	0.034	0.713	0.023	0.036
100	0.645	+- 0.019	0.031	0.702	0.020	0.032

The observed overhead ratios with standard deviation and 95% confidence levels are shown in Figure 42 and Table 37. Across the observed mobility range AODV produces less overhead than ACODV. At low mobility, the overhead of ACODV is at 11.9% only slightly higher than the 9.3% overhead of AODV. However as mobility is increased, the ACODV algorithm have to retransmit and backtrack more packets to achieve the improved delivery ratio described above, and hence the overhead of ACODV increases more with increasing mobility than the overhead of AODV. It is interesting to observe that the routing overhead curves are almost exact (inversed) replicas of the byte delivery curves. In the observed mobility range AODV produced from 2.5% (low mobility) to 9.7% (high mobility) less overhead than ACODV.

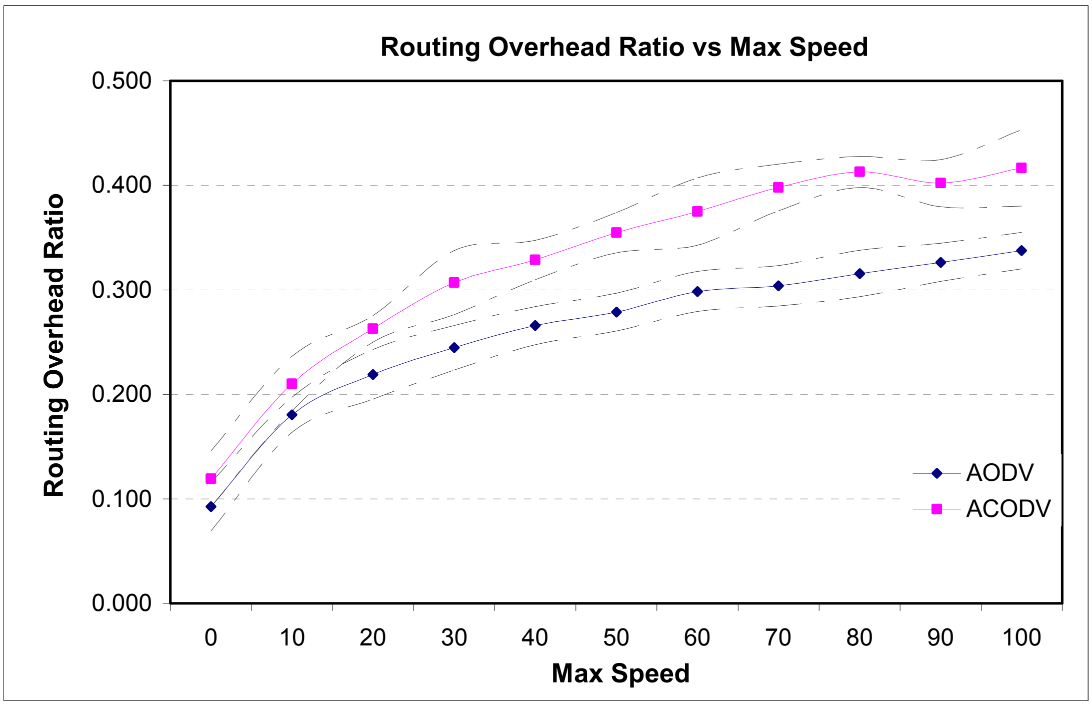


Figure 42 : Impact of mobility on routing overhead ratio

Table 37 : Impact of mobility on routing overhead ratio

	AODV		*ACODV*	
Max Speed	***Overhead Ratio***	***σ***	***Overhead Ratio***	***σ***
0	0.093 +- 0.023	0.038	0.119 +- 0.027	0.043
10	0.180 +- 0.017	0.027	0.210 +- 0.026	0.042
20	0.219 +- 0.024	0.038	0.263 +- 0.013	0.021
30	0.245 +- 0.021	0.034	0.307 +- 0.031	0.049
40	0.266 +- 0.018	0.030	0.329 +- 0.019	0.030
50	0.279 +- 0.018	0.029	0.355 +- 0.019	0.031
60	0.299 +- 0.019	0.031	0.375 +- 0.032	0.052
70	0.304 +- 0.019	0.031	0.398 +- 0.022	0.036
80	0.316 +- 0.022	0.036	0.413 +- 0.015	0.024
90	0.326 +- 0.018	0.029	0.402 +- 0.022	0.036
100	0.338 +- 0.017	0.028	0.417 +- 0.036	0.059

The observed end-to-end delays with standard deviation and 95% confidence levels are shown in Figure 43 and Table 38. When links fail, AODV reinitiates RREQ packets while ACODV retransmits or backtracks packets. It appears that the RREQ mechanism delays packets more than the retransmission/backtracking mechanism, as the end-to-end delay of AODV rises much more with higher mobility than the end-to-end delay of ACODV. It is also possible that the DARD mechanism causes RREQ packets to travel fewer hops so that route replies arrive quicker and packets can be delivered quicker. This was also observed when the DARD mechanism was evaluated in section 5.6. In the observed mobility range the end-to-end delay of ACODV increased by only 71% from 0.042 seconds to 0.072 seconds, while the delay of AODV increased by 184% from 0.050 seconds to 0.142 seconds.

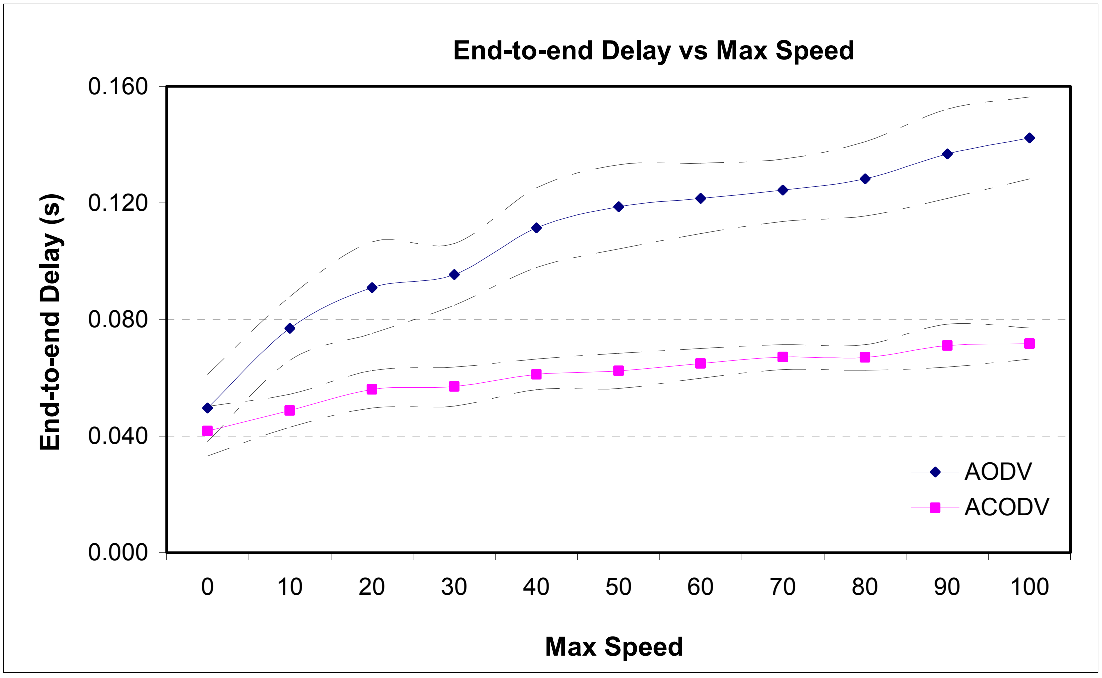


Figure 43 : Impact of mobility on end-to-end delay

Table 38 : Impact of mobility on end-to-end delay

	AODV			*ACODV*		
Max Speed	***End-to-end Delay***		***σ***	***End-to-end Delay***		***σ***
0	0.050	+- 0.012	0.019	0.042	+- 0.009	0.014
10	0.077	+- 0.011	0.018	0.049	+- 0.006	0.009
20	0.091	+- 0.016	0.025	0.056	+- 0.006	0.010
30	0.095	+- 0.011	0.017	0.057	+- 0.007	0.011
40	0.112	+- 0.014	0.022	0.061	+- 0.005	0.009
50	0.119	+- 0.014	0.023	0.062	+- 0.006	0.010
60	0.122	+- 0.012	0.019	0.065	+- 0.005	0.008
70	0.124	+- 0.011	0.017	0.067	+- 0.004	0.007
80	0.128	+- 0.013	0.021	0.067	+- 0.004	0.007
90	0.137	+- 0.015	0.025	0.071	+- 0.007	0.012
100	0.142	+- 0.014	0.023	0.072	+- 0.005	0.009

In summary, in the observed scenarios the ACODV algorithm provides:

- 5.5% to 10% increased byte delivery ratio,
- up to 49% (from 0.142s to 0.072s) reduced end-to-end delay at high mobility,

- at the cost of a routing overhead increase of 2.5% (low mobility) to 9.7% (high mobility).

6.3.2 Response to varying network load

When the packet load in a network is increased the probability of packet collisions and the time spent by packets in queues increases accordingly. The purpose of this section is to benchmark the performance of ACODV against AODV in such increasing load conditions.

To test the performance of these protocols in varying load conditions, experiments were created with increasing numbers of CBR source/destination pairs. In each experiment, random CBR source nodes sent 100 data packets to random destination nodes. The number of CBR sources was stepped from 1 to 100 in increments of 20, and for each number of sources the experiments were repeated with different random seed values, different random node starting positions and different application layers (CBR source/destination pairs). Because of the very long real-time required to run experiments with many CBR sources the experiments were only repeated 5 times. Experiments were repeated for AODV and ACODV with identical application layers to allow fair comparison. The experimental parameters are summarised in Table 39.

Table 39: Experimental setup for network load experiments

Parameter	***Value***
Number of nodes in network	100
Node initial placement	Random
Number of CBR sources	Stepped from 1 to 100 in 20 increments
CBR source start time	Random up to 180 seconds
Number of data packets created by each CBR source	100
Data packet size	1024 bytes
Terrain size	3000m x 1000m
Random Waypoint pause time (s)	1
Random Waypoint maximum speed (m/s)	10m/s
Number of times experiment repeated	5
General	Route reply by intermediate nodes enabled Backtracking enabled $p_{dard} = 0.25$ $\rho = 0.4$ $\varepsilon = 1$ second

The observed byte delivery ratios are shown in Figure 44 with standard deviation and 95% confidence intervals in Table 40. Confidence intervals are not shown in Figure 44 as the byte delivery ratio figures overlap too much to allow clear presentation of confidence intervals.

Both protocols show slightly higher byte delivery ratios with 20 CBR sources than with 1 source. This is due to the probability of a dropped packet being spread over more packets with 20 sources than with 1 source, resulting in a higher percentage of delivered packets. It also appears that the network load with 20 sources is not high enough to cause significant packet loss due to collisions and packet queues.

When the number of sources increases beyond 20, packet collisions and packet queues start to significantly impact on the delivery ratios of both protocols. Both protocols show a sharp drop in byte delivery ratios from 20 sources to 100 sources, with ACODV dropping by 60.9% from 88% to 27.1%, and AODV dropping by 58.2% from 77.5% to 19.3%.

Across the range of CBR sources ACODV delivered slightly more packets than AODV, with a maximum difference of 10.6% (1 source) and a minimum difference of 2.8% (80 sources). The improvements made to ACODV (multipath protocol, backtracking, DARD and the ACO algorithm) appears to make ACODV marginally better than AODV at delivering packets with increasing network load, although it still remains to be seen at what cost this improvement was realised.

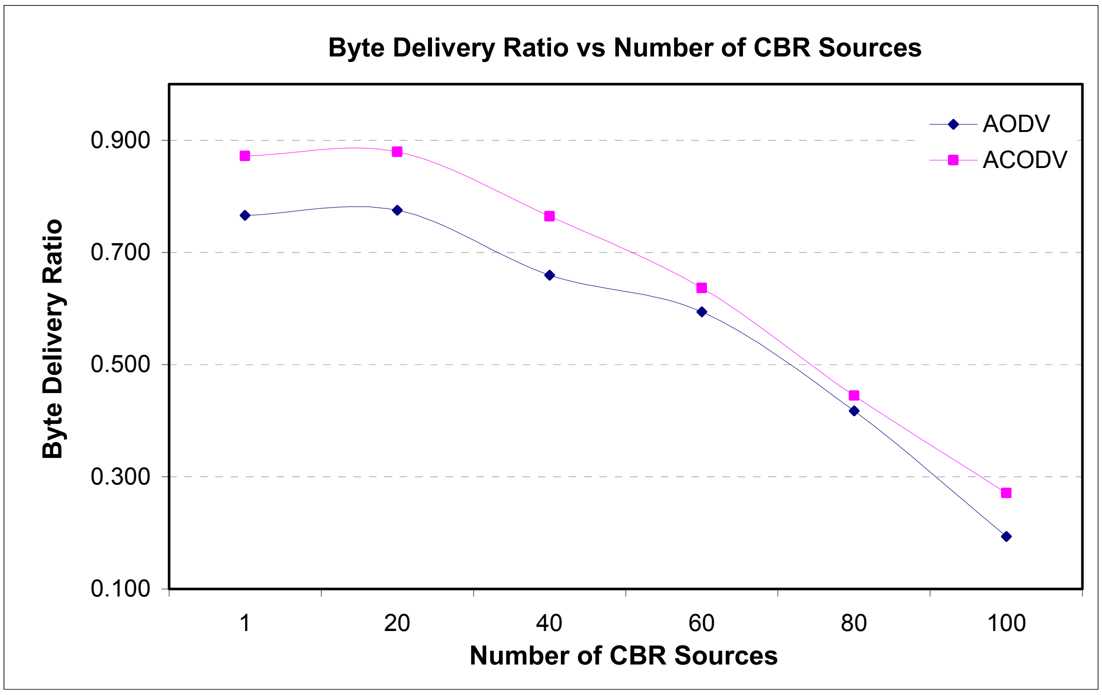


Figure 44 : Impact of network load on byte delivery ratio

Table 40 : Impact of network load on byte delivery ratio

	AODV			*ACODV*		
Number of CBR Sources	***Delivery Ratio***		***σ***	***Delivery Ratio***		***σ***
1	0.766	+- 0.059	0.095	0.872	+- 0.091	0.147
20	0.775	+- 0.016	0.026	0.880	+- 0.031	0.050
40	0.659	+- 0.018	0.029	0.764	+- 0.026	0.042
60	0.594	+- 0.026	0.043	0.636	+- 0.035	0.056
80	0.417	+- 0.014	0.022	0.445	+- 0.052	0.084
100	0.193	+- 0.013	0.021	0.271	+- 0.013	0.021

The observed routing overhead ratios are shown in Figure 45 with standard deviation and 95% confidence intervals in Table 41. Confidence intervals are not shown in Figure 45 as the routing overhead ratios figures overlap too much to allow clear presentation of confidence intervals. In all except the 100 source scenarios, ACODV produced slightly more overhead than AODV. With 1 source, ACODV produced 14.3% more overhead than AODV to maintain the 10.6% increase in byte delivery ratio observed earlier. In the 20, 40, 60 and 80 source scenarios AODV produced a minimum of 4.8% and an average of 6.7% less overhead than ACODV. The overhead ratios in both cases increase with increasing numbers of sources as the protocols have to compensate for increasing packet collisions and increasing time spent by packets in queues.

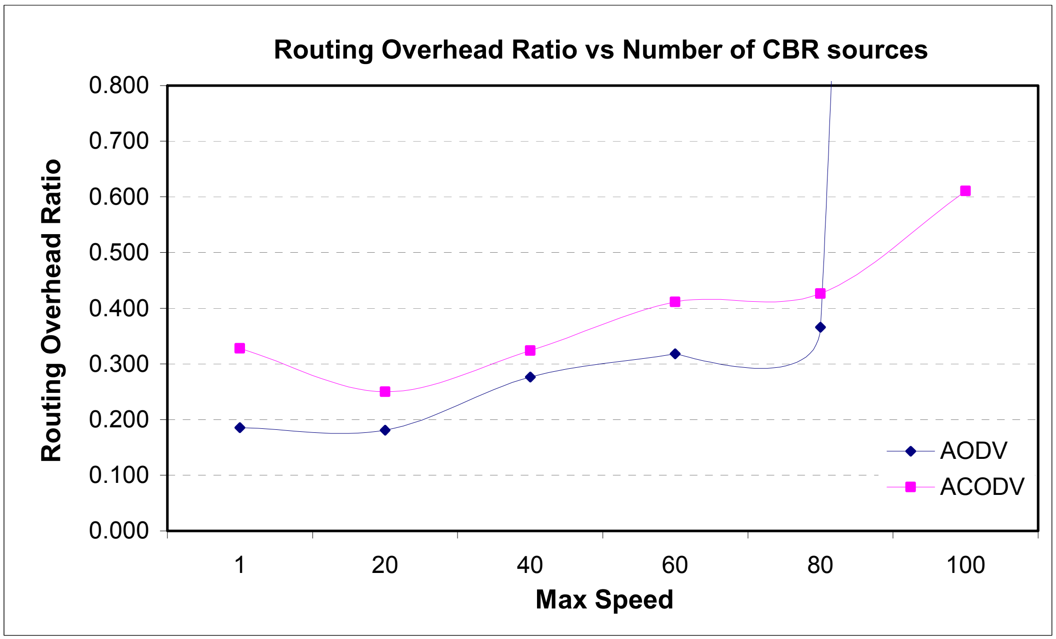


Figure 45 : Impact of network load on routing overhead ratio

Table 41 : Impact of network load on routing overhead ratio

	AODV			*ACODV*		
Number of CBR Sources	***Overhead Ratio***		***σ***	***Overhead Ratio***		***σ***
1	0.185	+- 0.098	0.157	0.328	+- 0.129	0.208
20	0.181	+- 0.013	0.021	0.250	+- 0.028	0.046
40	0.276	+- 0.019	0.031	0.324	+- 0.037	0.059
60	0.318	+- 0.025	0.041	0.411	+- 0.025	0.040
80	0.366	+- 0.006	0.010	0.426	+- 0.017	0.027
100	10.573	+- 0.368	0.594	0.610	+- 0.088	0.141

With 100 sources the overhead of AODV suddenly increases dramatically to 1057.3%, while the overhead of ACODV maintains its slow increase and only rises to 61%. It is not clear why there is such a sudden change from 80 to 100 sources and not a more gradual increase as one would expect. Analysis of the experimental data show that the number of hops traveled by RREQ and RREP packets, as well as the number of hops traveled by data packets, increases tenfold between 80 and 100 sources in AODV. It appears that the high network load is causing RREQ and RREP packets to travel much longer routes due to the unavailability (either from packet collisions or overloaded packet queues) of shorter routes. However, further experimental evaluation in future work should determine why this phenomenon becomes so suddenly apparent in AODV with 100 sources. This phenomenon is not present in ACODV due to the multipath nature of ACODV, and possibly due to the DARD mechanism continuously providing shorter routes to destinations.

6.3.3 Scalability of ACODV

As the size of real-life networks are constantly growing, the practical use of a routing protocol is often limited by the protocol's ability to scale reliably. The aim of this section is to evaluate the performance of ACODV in such growing networks.

The network setup used to test the scalability of ACODV is the same as the setup used to test the scalability of AODV [174] and AntHocNet [149]. In these studies, the number of nodes and the terrain size are varied in such a way that the average node density remains roughly constant, so that each node should have six to eight neighbours. The number of nodes and terrain sizes are shown in Table 42. However, in the AODV and AntHocNet experiments packets were generated until the end of the experiment, which implies that some packets will still be on their way to destination nodes at the experiment's end. In the experiments performed here, the number of packets sent is set to a fixed amount (100 per CBR source) to ensure that there are no packets remaining in-situ in the network when the simulation ends.

Table 42: Scalability experiments' network and terrain sizes

Number of Nodes	***Terrain Size (m)***
100	1500 x 1500
500	3500 x 3500
1000	5000 x 5000
1500	6000 x 6000

Di Caro, Ducatelle and Gambardella [149] introduced the very handy term 'computational restraints' to indicate that the machine on which experiments were performed could simply not handle any larger experiments. As a result of similar computational restraints and because of the very long real-time required to run large experiments, experiments were performed only for networks of up to 1500 nodes and were only repeated 3 times. An overview of the experimental parameters is given in Table 43.

Table 43: Experimental setup for scalability experiments

Parameter	***Value***
Experiment time	500 seconds
Number of CBR Sources	20
Number of data packets created by each CBR source	100
CBR source start time	Random up to 180 seconds
CBR source packet generation rate	0.25 seconds (4 packets per second)
Data packet size	512 bytes
Random Waypoint maximum speed	10 m/s
Random Waypoint pause time	30 seconds
General	Route reply by intermediate nodes enabled Backtracking enabled $p_{dard} = 0.25$ $\rho = 0.4$ $\varepsilon = 1$ second

The byte delivery ratios for AODV and ACODV are shown in Figure 46 and Table 44. Standard deviation and 95% confidence intervals are shown, although it has to be kept in mind that the significance of these statistical values is questionable with only 3 samples.

In the 100 node scenarios, ACODV slightly outperformed AODV, with both protocols delivering more than 90% of the routed packets. However, when the network grew to 500 nodes the delivery ratio of AODV fell by 25.3% to 70% while the delivery rate of ACODV only fell with 2% to 97%. With network size increasing to 1500 nodes the delivery rate of AODV drops to 44.2% while ACODV still manages to deliver 84% of routed packets.

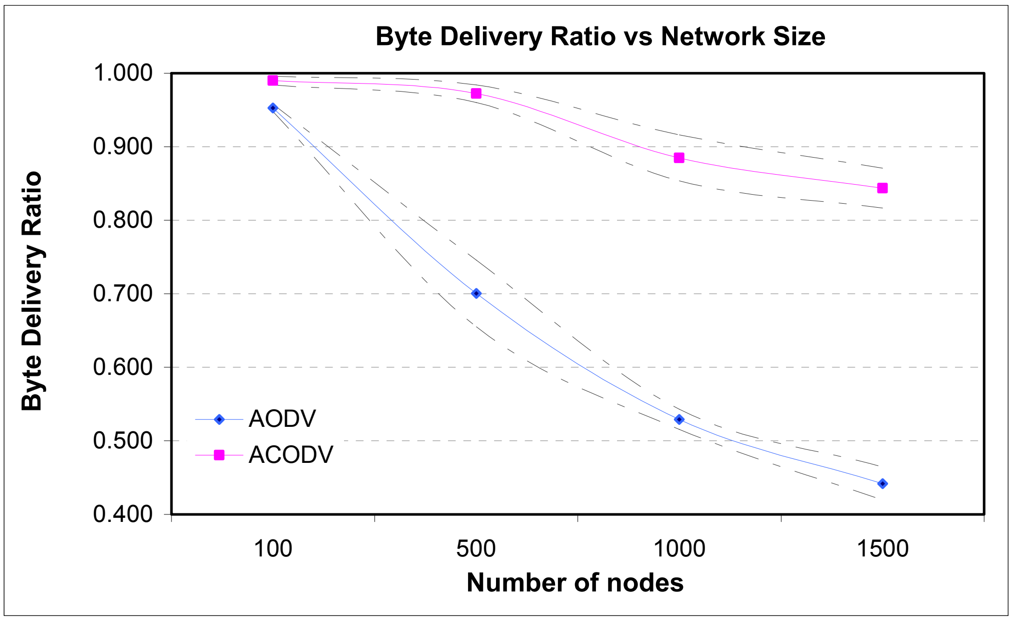


Figure 46 : Impact of network size on byte delivery ratio

Table 44 : Impact of network size on byte delivery ratio

	AODV			*ACODV*		
Number of Nodes	***Delivery Ratio***		***σ***	***Delivery Ratio***		***σ***
100	0.953	+- 0.005	0.008	0.99	+- 0.006	0.010
500	0.700	+- 0.045	0.073	0.97	+- 0.012	0.019
1000	0.529	+- 0.014	0.022	0.88	+- 0.031	0.051
1500	0.442	+- 0.022	0.036	0.84	+- 0.027	0.044

The byte delivery ratios of AODV described above is similar to the byte delivery ratios for AODV reported in the scalability experiments performed by Di Caro, Ducatelle and Gambardella [149]. The large difference in delivery ratio between AODV and ACODV can be attributed to various factors:

- Most notably, AODV is a single route algorithm. When a single link failure occurs the upstream node may do local repair if the destination is within a certain hop distance [51], by default set to 0.3·*network_diameter*. Any link failure occurring outside of this boundary immediately results in the packet being dropped. As the network size grows, the chances of link failures occurring outside of AODV's local repair boundary increase and a single link failure regularly results in a packet being dropped. ACODV maintains multiple routes to the destinations and on the failure of a link retries to send the packet along other routes.
- When all attempts by an intermediate node in ACODV to forward a packet fails, the packet is backtracked. Each upstream node potentially has multiple routes to the destination, increasing the probability of the packet being delivered. The packet is only

dropped when it is backtracked back to the source node. This feature is not supported by AODV.

- As a result of the stochastic forwarding of packets in ACODV some exploration of alternative routes occurs naturally, thereby increasing the probability of a viable alternative route existing on link failure. Being a single route algorithm, AODV does not have this benefit.
- It is possible that forwarding of packets based on pheromone (ACODV) in stead of hopcount (AODV) results in more reliable routes being used as pheromone accumulates on frequently used links.
- The dynamics of the route timeout mechanism using pheromone in ACODV is quite different from the timer mechanism used in AODV. In AODV a route's timeout is incremented by a fixed period when used, and route lifetime decreases linearly. In ACODV the pheromone deposited when a route is used is a function of hopcount, and pheromone decreases exponentially. It is possible that the differing dynamics provide improved control over the removal of old routes.

The observed routing overhead ratios for AODV and ACODV are shown in Figure 47 and Table 45 with standard deviation and 95% confidence intervals. The routing overhead ratio of AODV increases roughly linearly with increasing network sizes, and is consistent with the control overhead reported for AODV by Lee, Belding-Royer and Perkins in [174]. In the 100-node scenarios, the overhead of ACODV is at 21% much higher than the 7% overhead of AODV. This is probably due to retransmissions and backtracking necessary to maintain the slightly higher delivery ratio of ACODV (99% compared to 95.3% for AODV) in the 100-node scenarios.

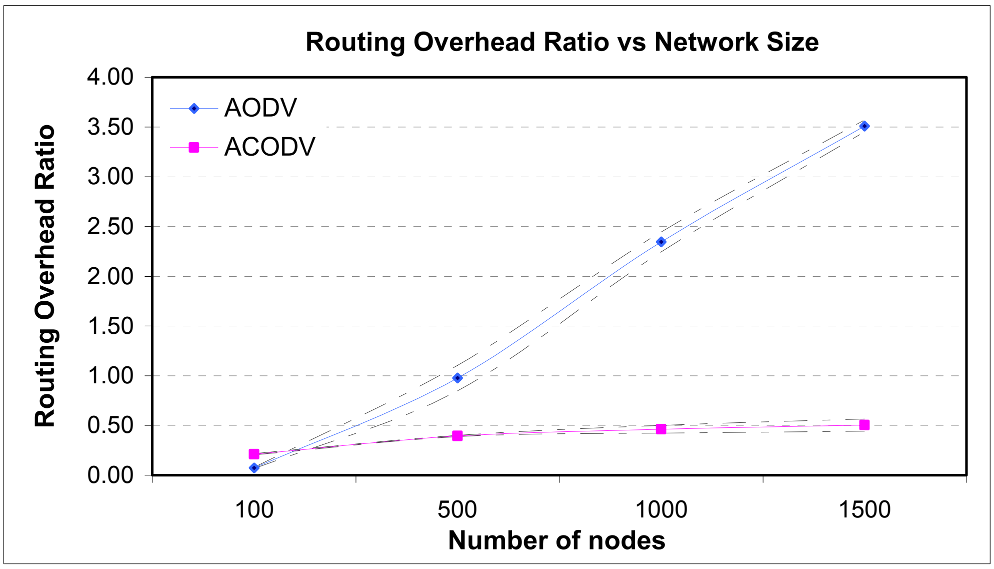


Figure 47 : Impact of network size on routing overhead ratio

Table 45 : Impact of network size on routing overhead ratio

	AODV			***ACODV***		
Number of Nodes	***Overhead Ratio***		***σ***	***Overhead Ratio***		***σ***
100	0.07	+- 0.005	0.009	0.21	+- 0.006	0.009
500	0.98	+- 0.128	0.207	0.40	+- 0.007	0.012
1000	2.34	+- 0.099	0.160	0.46	+- 0.039	0.062
1500	3.51	+- 0.059	0.095	0.51	+- 0.062	0.099

As network sizes increase, there is a dramatic difference between the overhead ratios of AODV and ACODV. With network size increasing from 500 to 1500 nodes the overhead of AODV increases by 258% from 98% to 351%. In the same interval the overhead of ACODV only increases by 142% from 21% to 51%.

The significant difference in routing overhead ratios can again be attributed to various factors:

- There is a subtle but noteworthy difference in the RERR mechanisms used by AODV and ACODV. In AODV, *any* link failure results in a RERR packet being sent along the entire route back to the source node. Now consider a link failure at node *i* in the ACODV network where *i* is trying to forward a packet to destination *d* through next-hop *n*. If *i* have alternative routes to *d* and *n* available, then no RERR will be sent. If ***i*** have alternative routes to *d* but not to *n*, a RERR will be sent to upstream node *p* listing *only n* as unreachable. If upstream node *p* does not have a route to *n* (since *n* is an intermediate node on the route to *d*) then from node *p*'s perspective no destinations became unreachable, and *p* will not forward the RERR packet. If node *i* does not have any alternative routes to destination *d* it will send a RERR listing *d* as unreachable. This RERR will only be forwarded by upstream nodes if they also do not have any alternative routes to destination *d*. The propagation of RERR packets in ACODV is therefore greatly reduced compared to AODV.
- When link failures occur in AODV, RREQ packets are initiated either by the source or an intermediate node. When link failures occur in ACODV the algorithm retries the packet along alternative routes, and if no alternatives exist backtracks the packet. The routing overhead of a RREQ packet flooded through the network is considerably more than that of a retried or backtracked packet.
- After an initial link from source to destination is established, any link failure in AODV results in the flooding of RREQ packets and resulting RREP packets. In ACODV, the DARD mechanism (with its curious characteristic of not significantly increasing overhead) constantly seeds the network with routes to the destination. The probability of a RREQ having to be reinitiated is therefore greatly reduced. Additionally, if a RREQ is reinitiated the RREQ travels fewer hops in search of a route to the destination as more intermediate

nodes have been seeded with routes to the destination.

The ability of ACODV to maintain reasonably high byte delivery ratios with low overhead in large networks makes it a suitable algorithm for networks with more than 1000 nodes. However this increased performance comes at the cost of a higher computational requirement necessary for maintaining pheromone levels and performing probability calculations, and at the cost of a higher memory requirement necessary for maintaining multiple routes to destinations.

6.4 Functions not supported by ACODV

The following functions may be implemented in future work but is not currently supported by ACODV:

- Priority routing of packets, all packets are currently routed on first-come-first-serve basis;
- Local repair of routes as well as query localization [175] is possible in ACODV and has been shown to improve a protocol's performance [174], but has not been implemented yet;
- Quality of Service (QoS) routing;
- Mobility awareness of nodes to dynamically adjust routing parameters;
- Power awareness of nodes to avoid nodes with low battery power; and
- Congestion awareness of nodes to re-route traffic through less congested nodes.

6.5 Conclusion

This chapter introduced the ACODV routing protocol for low-power ad hoc sensor networks. The configuration parameters as well as the structure of the routing table was described, followed by details on the generation, format and processing of RREQ, RREP, RERR and DERR packets.

Simulations of the ACODV protocol in different mobility conditions indicated that ACODV delivers up to 10% more packets than AODV in high mobility scenarios while reducing end-to-end delay by up to 49%. Of course this improved performance does not come for free, ACODV produces up to 9.7% higher routing overhead at high mobility and has much higher processing and memory requirements than AODV.

Evaluation of the performance of ACODV under different network load conditions revealed that ACODV delivers 2.8% to 10.6% more packets than AODV, again at the cost of 4.8% to 14.3% more overhead. Both ACODV and AODV do not cope overly well with large numbers of data sources, the byte delivery ratios dropped to 27.1% and 19.3% for ACODV and AODV respectively in 100 source scenarios. A strange phenomenon was observed in AODV where the overhead ratio increases tenfold with 100 sources, apparently due to packets being forced to travel longer routes. This phenomenon was not observed in ACODV, probably because of the multipath nature of ACODV and the DARD mechanism continuously providing shorter routes to destinations.

The real potential of ACODV was revealed in the scalability experiments. The delivery ratio of AODV dropped to 44.2% in networks with 1500 nodes, with ACODV managing to deliver 84% of the packets in the same networks. In addition, the overhead ratio of AODV rises almost linearly with increasing network size, rising to 351% in 1500 node networks, while the overhead ratio of ACODV only rises to 51% in the same networks. In other words, in large networks ACODV delivers roughly twice as many packets as AODV while producing seven times less overhead.

Although much more development on ACODV has to be done the protocol in its current form is fairly mature, and the description provided in this work should be sufficient to allow independent implementation of ACODV in a range of real-life applications.

7 Conclusion and Future work

This short chapter summarises the findings of this work. In addition, future work resulting from these findings is suggested.

7.1 Conclusion

The aim of this work was to define and evaluate a routing protocol suitable for very low-powered sensor networks. Existing routing protocols were overviewed, and the AODV protocol was seen to have the most promising features. Two prominent features of AODV that makes it suitable for low-power environments are: a. the use of distance-vector routing which usually generates less routing overhead than source routing; and b. the ability to function as a purely reactive protocol.

Swarm intelligence was then overviewed, and it was found that the ACO algorithm exhibits features which are highly desirable in an ad hoc routing protocol. The attractiveness of swarm intelligence in ad hoc routing protocols was confirmed when protocols that employ swarm intelligence were reviewed. Almost all works reviewed reported superior performance after the introduction of swarm intelligence.

Metrics used for evaluating a protocol's performance was reviewed next. The metrics were classified as either scenario metrics, which describe the environment wherein a protocol operates, or performance metrics, which describe the actual performance of the protocol in that environment. In addition, specific features of a protocol which reveal more about the protocol's operation and which are relevant to network designers were reviewed.

The definition of the routing overhead ratio metric was investigated and it was found that many works have different definitions of this metric. In addition, even if a standard definition is used, different interpretations of this definition could result in differences of up to 72% in reported results. It was decided that a strict interpretation of the metric's definition yielded the most appropriate results, and all further results in this work was reported using a strict interpretation.

The backtracking mechanism was then investigated, and it was found that the introduction of this mechanism yielded byte delivery ratio improvements of up to 16.3% at high node mobility. This improvement was realised at the cost of an increase in routing overhead of roughly 55%, and an increase in end-to-end delay of up to 23.8% at high node mobility. It was also found that the higher end-to-end delay due to backtracking increases with higher node mobility, while the increase in routing overhead stays roughly constant.

Investigation of the RREQ mechanism revealed that the sending of more RREQ packets does not necessarily lead to more routes being discovered. If RREQ packets are sent with insufficient broadcast jitter, packet collisions cause a high number of RREQ packets to be lost and less of the network to be discovered. However if excessive broadcast jitter is used, the long delays may result in routes being stale by the time they are used. It was established that broadcast jitter needs to be carefully tuned to a specific environment.

The RREP mechanism was shown to be an inadequate mechanism for finding multiple routes to destinations. Even in the simplest scenario of a 25-node stationary network where all nodes have a degree of connectivity of at least 3, only an average of 60% of initiated RREP packets reached their intended source nodes. Introducing some mobility in the same network slightly increased the percentage of successfully delivered RREP packets to 67%. Since the ACO algorithm relies on the RREQ/RREP mechanism to provide multiple routes to destinations, a mechanism was needed to increase the number of routes to destinations available throughout the network. A mechanism called destination assisted route discovery (DARD) was introduced which actively involves destination nodes in the route finding process. The mechanism was evaluated and it was found that DARD improved byte delivery ratios on average by 8% without increasing routing overhead.

An ACO algorithm similar to SACO was then introduced to make next-hop decisions. Experimental evaluation of the effects of different pheromone evaporation rates revealed that the algorithm is not overly sensitive to slightly suboptimal values, and that a setting can be found which provides satisfactory performance across the 0-100m/s mobility range.

Evaluation of the effects of different pheromone amplification factors yielded surprising results. The pheromone amplification factor was shown not to have any significant impact on byte delivery ratios or routing overhead ratios. Further investigation revealed a simple reason behind this - nodes rarely have more than one route to a destination. Even with the DARD mechanism active, in 66% of the cases where a next-hop decision had to be made only one route was available. In another 20.1% of the cases only 2 routes were available, and with no guarantee of node or link disjointness these routes may very well only differ by one or two hops. In only 13.9% of the cases did a node have 3 or more choices. Mechanisms were suggested which could increase the number of routes available at nodes so that the ACO algorithm may be used more effectively.

After the investigation of routing mechanisms an algorithm suitable for low power environments was defined, namely the Ant Colony Optimisation Distance Vector (ACODV) routing protocol. The operation of this protocol was described in detail.

Comparison of the performance of ACODV and AODV in networks with varying node mobility showed that ACODV delivered 5% to 10% more packets than AODV, but at the cost of 2.5% (low mobility) to 9.7% (high mobility) more overhead than AODV. The performance of ACODV under different network load conditions was investigated next, and experimental results indicated that ACODV delivers 2.8% (1 CBR source) to 10.6% (100 CBR sources) more packets than AODV at the cost of 4.8% (1 CBR source) to 14.3% (90 CBR sources) more overhead. Both ACODV and AODV was seen not to cope very well with large numbers of data sources as the byte delivery ratios dropped to 27.1% and 19.3% for ACODV and AODV respectively in 100 source scenarios. Scalability experiments in networks of up to 1500 nodes indicated that ACODV delivers up to 39.8% more packets than AODV in large networks while producing a seventh (51% compared to 351%) of the routing overhead of AODV.

7.2 Future work

Consolidation

Although the past decade has seen an explosion of ad hoc related research, the same cannot be said for real-life implementations of ad hoc networks. To date there are hardly any operational ad hoc networks, and the few existing examples usually employ less than 50 nodes. There exists a plethora of ad hoc related works on all aspects of ad hoc networks from the physical layer up to network and application layers. It is the opinion of this author that future ad hoc work should focus on consolidating these works, using best-of-class at each layer and assembling complete, usable products.

Acquiring more routes

The pheromone amplification experiments in section 5.7.2 revealed that the effectiveness of the ACO algorithm employed in ACODV is greatly reduced by the fact that nodes usually have only one or two routes to a destination available. Future work should attempt to increase the number of routes available at nodes so that the ACO algorithm can reach its full potential. This can be done by enabling local repair of routes at intermediate nodes. If a parameter is introduced that specifies the minimum number of routes to an active destination, then intermediate nodes can use the local repair mechanism to find the required number of routes to that destination. Query localization [175] may also be introduced to limit the flooding of RREQ packets from nodes seeking multiple routes.

Power-aware and Congestion-aware routing

Many recent works [158, 176] have included power-management in the routing process. Since the nodes in a sensor network are battery powered the routing load should be spread through all nodes in the network so that individual nodes are not overloaded. Battery levels and congestion levels (which indicate battery drain) may be included in the pheromone depositing process, so that these factors are taken into regard when next-hop decisions are made. Future work may also investigate the effects of including the battery and congestion levels of an entire route in the pheromone depositing process.

It has been shown [176] that larger packet sizes increase power efficiency, as the packet overhead is amortised over more bits. Future work should attempt to find mechanisms for aggregating packets so that this effect is exploited.

ACO Parameter Tuning

The amount of pheromone deposited in this work is proportional to 1/*hopcount* (refer to equation (5.5)). This means that 50% of the deposited pheromone decays in the first hop. A more suitable approach may be to have 50% of the pheromone decay in the first 50% of the network, so that the ACO algorithm is less biased towards short routes. Future work may also investigate the effects of making nodes mobility-aware so that ACO parameters can be dynamically adjusted according to node mobility.

Neural ACO

Although not strictly related to this work, consider an optimisation problem with a large number of inputs, such as selecting an optimum portfolio of shares. An ensemble of neural networks, each with different inputs, selects candidate shares. An ACO algorithm is then used to iteratively build a portfolio of shares from the candidates provided by the neural ensemble, similar to the ACO approached to solve TSPs. Pheromone is deposited as a function of the performance of each selected portfolio. As the underlying real-world inputs change over time different neural networks will select high-performance shares. This dynamic topology can be seen as analogue to the dynamic topologies encountered in ad hoc networks. It is hoped that the ACO algorithm will adapt and continue to find optimal portfolios just as the algorithm adapts to find optimal routes.

Bibliography

[1] J. Jubin and J.D. Tornow, "The DARPA Packet Radio Network Protocols", *Proceedings of the IEEE*, vol. 75, pp. 21-32, 1987.

[2] B.M. Leiner, D.L. Nielson, and F.A. Tobagi, "Issues in Packet Radio Network Design", *IEEE Special issue on Packet Radio Networks*, vol. 75(1), pp. 6-20, 1987.

[3] Wireless World Research Forum (WWRF), http://www.ist-wsi.org, *accessed* 2005.

[4] The Official IETF MANET Working group webpage, http://www.ietf.org/html.charters/manet-charter.html, *accessed* 2004.

[5] P. Varaiya, "Smart Cars on Smart Roads: Problem of Control", *IEEE Transactions on Automation and Control*, vol. 38, pp. 195-207, 1993.

[6] J.A. Freebersyser and B.M. Leiner, "A DoD perspective on mobile ad hoc networks", in *Ad Hoc Networking*, C. Perkins, Ed.: Addison Wesley, pp. 29-51, 2001.

[7] K. Akkaya and M. Younis, "A Survey on Routing Protocols for Wireless Sensor Networks", *Ad Hoc Networks*, vol. 3, pp. 325-349, 2004.

[8] I.F. Akyildiz, "Wireless sensor networks: a survey", *Computer Networks*, vol. 38(4), pp. 39-422, 2002.

[9] IEEE, "Standard for Wireless LAN - Medium Access control and Physical Layer Specification, P802.11/D10", *Technical Report*, 1999.

[10] S. Corson, J. Maker, and J.H. Cernicione, "Internet-based mobile ad hoc networking", *IEEE Internet Computing*, vol. 3, pp. 63-70, 1999.

[11] C.F. Chiasserini and R.R. Rao, "Pulsed battery discharge in communication devices", presented at The Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOMM '99), Seattle, USA, pp. 88-95, August 1999.

[12] E. Oida and M. Sekido, "An agent-based routing system for QoS guarantees", presented at IEEE International Conference on Systems, Man, and Cybernetics, pp. 833-838, Oct 1999.

[13] T. White, B. Pagurek, and A. Bieszczad, "Mobile agents for network management", *IEEE Communication Surveys*, vol. 1, 1998.

[14] E. Bonabeau, M. Dorigo, and G. Theraulaz, "Swarm Intelligence: From Natural to Artificial Systems", *Artificial Life*, vol. 7, pp. 315-319, 2001.

[15] M. Ward, "There's an ant in my phone", *New Scientist*, pp. 32, January 1998.

[16] S. Giordano, "Mobile ad hoc networks", in *Handbook of Wireless Networks and Mobile Computing*, I. Stojmenovic, Ed. New York: Wiley, pp. 325-346, 2002.

[17] M. Dorigo and G. Di Caro, "The ant colony optimization meta-heuristic", in *New Ideas in Optimization*, D. Corne, M. Dorigo, and F. Glover, Eds. London, UK: McGraw-Hill, pp. 11-32, 1999.

[18] S. Keshav, "An Engineering Approach to Computer Networking: ATM networks, the Internet, and the Telephone Network, Chapter 11": Addison Wesley, pp. 287-357, 1997.

[19] Xiao Yan Hong, Kai Xin Xu, and Mario Gerla, "Scalable Routing Protocols for Mobile Ad Hoc Networks", *IEEE Network*, vol. July/August 2002, pp. 11-21, 2002.

[20] A. Udaya Shankar, C. Alaettinoglu, I. Matta, and K. Dussa-Zieger, "Performance comparison of routing protocols using MaRS: distance-vector versus link-state", presented at ACM Sigmetrics and Performance, International Conference on Measurement and Modeling of Computer Systems, Newport, USA, pp. 181, June 1992.

[21] C.C. Chiang, "Routing in Clustered Multihop Mobile Networks with Fading Channel", presented at IEEE SICON, pp. 197-211, April 1997.

[22] Chien Chung Shen, Chavalit Srisathapornphat, Rui Liu, Zhou Chuan Huang, Chaiporn Jaikaeo, and Errol L. Loyd, "CLTC: A Cluster-based Topology Control Framework for AdHoc Networks", *IEEE Transactions on Mobile Computing*, vol. 1, pp. 18-32, 2004.

[23] Mehran Abolhasan, Tadeusz Wysocki, and Eryk Dutkiewicz, "A Review of Routing Protocols for Mobile Ad Hoc Networks", *Ad Hoc Networks*, vol. 2, pp. 1-22, 2004.

[24] E. Royer and Chai Keong Toh, "A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks", *IEEE Personal Communications*, pp. 46-55, 1999.

[25] D. Lang, "A Comprehensive Overview About Selected Ad Hoc Networking Routing Protocols", *Technical Report,* Technishe Universität München, München, Germany, March 2003.

[26] L.M. Feeney, "A Taxonomy for Routing Protocols in Mobile Ad Hoc Networks", *Technical Report,* Swedish Institute of Computer Science, Krista, Sweden, October 1999.

[27] J.J. Garcia-Luna-Aceves and C. Marcelo Spohn, "Source-tree Routing in Wireless Networks", presented at Seventh Annual International Conference on Network Protocols, Toronto, Canada, pp. 273, October 1999.

[28] K.K. Kasera and R. Ramanathan, "A Location Management Protocol for Hierarchically Organised Multihop Mobile Wireless Networks", presented at IEEE ICUPC'97, San Diego, CA, pp. 158-162, October 1997.

[29] G. Pei, Mario Gerla, Xiao Yan Hong, and C.C. Chiang, "A Wireless Hierarchical Routing Protocol with Group Mobility", presented at Wireless Communications and Networking Conference, New Orleans, USA, pp. 1536-1540, September 1999.

[30] M. Jiang, J. Ji, and Y.C. Tay, "Cluster Based Routing Protocol", *IETF Internet Draft, draft-ietf-manet-cbrp-spec-02.txt*, 1999.

[31] M. Joa-Ng and I.T. Lu, "A Peer-to-Peer Zone-based Two-level Link State Routing for Mobile Ad Hoc Networks", *IEEE Journal on Selected Areas of Communications*, vol. 17, pp. 1415-1425, August 1999.

[32] S.C. Woo and S. Singh, "Scalable Routing Protocol for Ad Hoc Networks", *Wireless Networks*, vol. 7, pp. 513-529, 2001.

[33] H. Takagi and L. Kleinrock, "Optimal Transmission Ranges for Randomly Distributed Packet Radio Terminals", *IEEE Transactions on Communications*, vol. 32(3), pp. 246-257, 1984.

[34] P. Larsson, "Selection Diversity Forwarding", presented at Mobile Ad Hoc Networking and Computing (MobiHoc), Long Beach, USA, pp. 279-282, October 2001.

[35] N. Nikaein, H. Laboid, and C. Bonnet, "Distributed Dynamic Algorithm (DDR) for Mobile Ad Hoc Networks", presented at MobiHOC 2000: First Annual Workshop on Mobile Ad Hoc Networking and Computing, Boston, USA, pp. 19-27, 2000.

[36] S. Radhakrishnan, N.S. Rao, G. Racherla, C.N. Sekharan, and S.G. Batsell, "A Routing Protocol for Ad Hoc Networks Using Distributed Spanning Trees", presented at IEEE Wireless Communications and Networking Conference, New Orleans, USA, pp. 100-104, 1999.

[37] N. Nikaein and C. Bonnet, "HARP - Hybrid Ad Hoc Routing Protocol", presented at International Symposium on Telecommunications (IST), Tehran, Iran, pp. unknown, September 2001.

[38] M. Gerla, "Fisheye State Routing Protocol (FSR) for Ad Hoc Networks", *IETF Internet Draft, draft-ietf-manet-fsr-03.txt*, June 2002.

[39] C. Santivanez, R. Ramanathan, and I. Stavrakakis, "Making Link State Routing Scale for Ad Hoc Networks", presented at 2001 ACM International Symposium on Mobile Ad Hoc Networking & Computing, MobiHOC'2001, Long Beach, USA, pp. 22-32, October 2001.

[40] P. Jacquet, P. Mishlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot, "Optimized Link State Routing Protocol for Ad Hoc Networks", presented at IEEE International Multi Topic Conference, pp. 62-68, 2001.

[41] R. Ogier, F. Templin, and M. Lewis, "Topology Dissemination Based on Reverse-Path Forwarding (TBRPF)", *IETF Internet Draft, draft-ietf-manet-tbrpf-06.txt*, 2003.

[42] Y. Dalal and R. Metclafe, "Reverse Path Forwarding of Broadcast Packets", *Communications of the ACM*, vol. 21(12), pp. 1040-1048, December 1978.

[43] E. Dijkstra, "A Note on Two Problems in Connection with Graphs", *Numerishe Mathematik*, vol. 1, pp. 269-271, 1959.

[44] C. Perkins and T.J. Watson, "Highly Dynamic Destination Sequenced Distance Vector Routing (DSDV) for Mobile computers", presented at ACM SIGCOMM'94 Conference on Communications Architectures, London, UK, pp. 234-244, 1994.

[45] S. Murthy and J.J. Garcia-Luna-Aceves, "A Routing Protocol for Packet Radio Networks", presented at First Annual ACM International Conference on Mobile Computing and Networking, Berkeley, USA, pp. 86-95, 1995.

[46] J.J. Garcia-Luna-Aceves and J. Raju, "A Comparison of On-demand and Table Driven Routing for Ad Hoc Wireless Networks", presented at ICC 2000 - IEEE International Conference on Communications, pp. 1702-1706, 2000.

[47] J.J. Garcia-Luna-Aceves and J. Raju, "Scenario-based Comparison of Source-tracing and Dynamic Source Routing Protocols for Ad Hoc Networks", *ACM Computer Communication Review - Special Issue on Mobile Extensions to the Internet*, pp. 70-81, October 2001.

[48] S. Basagni, I. Chlamtac, V.R. Syrotivk, and B.A. Woodward, "A Distance Effect Algorithm for Mobility (DREAM)", presented at Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCOM'98), Dallas, USA, pp. 76-84, 1998.

[49] Y.B. Ko and N.H. Vaidya, "Location-aided Routing (LAR) in Mobile Ad Hoc Networks", *Wireless Networks*, vol. 6(4), pp. 307-321, July 2000.

[50] J.M. McQuillan, I. Richer, and E.C. Rosen, "The new routing algorithm for the ARPANET", *IEEE transactions on Communications*, vol. COM-28, pp. 711-719, 1980.

[51] S.R. Das, C. Perkins, and E. Royer, "Ad Hoc On Demand Distance Vector (AODV) Routing", *IETF Internet Draft, draft-ietf-manet-aodv-13.txt*, 2003.

[52] S.R. Das, C. Perkins, and E. Royer, "Performance Comparison of Two On-demand Routing Protocols for Ad Hoc Networks", presented at InfoCom, pp. 3-12, 2000.

[53] M.K. Marina and S.R. Das, "On-Demand Multipath Distance Vector Routing for Ad Hoc Networks", presented at International Conference for Network Protocols (ICNP), Riverside, USA, pp. 14-23, 2001.

[54] S. Mueller and D. Ghosal, "Multipath Routing in Mobile Ad Hoc Networks: Issues and Challenges", presented at Modeling, Analysis, and Simulation On Computer and Telecommunications Systems (MASCOTS), Orlando, USA, pp. 209-234, 2003.

[55] D. Johnson, D. Maltz, and J. Broch, "DSR - The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks", in *Ad Hoc Networking*, C. Perkins, Ed.: Addison-Wesley, pp. 139-172, 2001.

[56] R.C. Dixon and D.A. Pitt, "Addressing, bridging, and source routing", *IEEE Network*, vol. 2, pp. 25-32, 1988.

[57] M. FransKaashoek, R. van Renesse, H. van Staveren, and A.S. Tanenbaum, "FLIP: An internetwork protocol for supporting distributed systems", *ACM Transactions on Computer Systems*, vol. 11, pp. 73-106, 1993.

[58] V. Park and S. Corson, "A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks", presented at IEEE INFOCOM '97, Kobe, Japan, pp. 1405-1413, April 1997.

[59] V. Park and S. Corson, "Temporally-Ordered Routing Algorithm (TORA)", *IETF Internet Draft, draft-ietf-manet-tora-03.txt*, June 2001.

[60] E.M. Gafni and D.P. Bertsekas, "Distributed Algorithms for Generating Loop-Free Routes in Networks With Frequently Changing Topology", *IEEE Transactions on Communications*, vol. COM-29, pp. 11-18, 1981.

[61] S. Corson and A. Ephremides, "A Distributed Routing Algorithm for Mobile Wireless Networks", *ACM/Baltzer Wireless Networks*, vol. 1(1), pp. 61-81, February 1995.

[62] Chai Keong Toh, "A Novel Distributed Routing Protocol to Support Ad Hoc Mobile Computing", presented at IEEE 15th Annual International Phoenix Conference on Computers and Communications, pp. 480-486, 1996.

[63] R.K. Padmanaban, V.R.S. Manikandan, and R. Naveenan, "Optimized Associativity-based Threshold Routing for Mobile Ad Hoc Networks", presented at International Conference on High Performance Computing (HiPC), Hyderabad, pp. unknown, December 2001.

[64] R. Dube, C. Rais, K. Wang, and S. Tripathi, "Signal Stability Based Adaptive Routing (SSA) for Ad Hoc Mobile Networks", *IEEE Personal Communications*, vol. 4(1), pp. 36-45, 1997.

[65] Z.J. Haas and R. Pearlman, "Zone Routing Protocol for Ad Hoc Networks", *IETF Internet Draft, draft-ietf-manet-zrp-02.txt*, 1999.

[66] E. Marais, "The Soul of the White Ant, originally published in Afrikaans as "Die Siel van die Mier", translated by Winifred de Kok", 1971.

[67] M. Maeterlinck and A. Sutro, *The Life of the White Ant*. New York: Dodd, Mead, 1927.

[68] Stigmergy, http://institute.advancedarchitecture.org/Research/Ants/Stigmergy, *accessed* 2004.

[69] E. Marais, *Die Siel van die Mier (The Soul of the Ant)*. Pretoria, South Africa: J.L. van Schaik, 1948.

[70] P.-P. Grasse, "La Reconstruction du nid et les coordinations interindividuelles: La theorie de la stigmergie", *Insectes Sociaux*, vol. 6, pp. 41-84, 1959.

[71] R. Schoonderwoerd, O. E. Holland, J. L. Bruten, and L. J. M. Rothkrantz, "Ant-Based Load Balancing in Telecommunications Networks", *Adaptive Behavior*, vol. 5, pp. 169-207, 1996.

[72] E.O. Wilson, *Sociobiology*. Cambridge, USA: Harvard University Press, 1975.

[73] E. Bonabeau, G. Theraulaz, V. Fourcassie, and Deneubourg J.L., "The Phase-Ordering Kinetics of Cemetery Organization in Ants", *Physical Review E*, vol. 57, pp. 4568-4571, 1998.

[74] Y.U. Cao, A.S. Fukunaga, and A.B. Kahng, "Cooperative Mobile Robotics: Antecedents and Directions", *Autonomous Robots*, vol. 4, pp. 1-23, 1997.

[75] R. Dorf and N. Shimon, *Concise International Encyclopedia of Robotics: Applications and Automation*: John Wiley & Sons, 1990.

[76] A.H. Bond and L. Gasser, *Readings in Distributed Artificial Intelligence*: Morgan Kaufmann Publishers, 1988.

[77] I. Kassabalidis, M.A. El-Sharkawi, R.J. Marks, P. Arabshahi, and A.A. Gray, "Swarm Intelligence for Routing in Communication Networks", *Technical Report,* Nasa Jet Propulsion Laboratory, 2001.

[78] T. Vicsek, A. Czirok, E. Ben-Jacob, I. Cohen, and O. Shochet, "Novel type of phase transition in a system of self-driven particles", *Physical Review Letters*, vol. 75, pp. 1226, 1995.

[79] E. Bonabeau, G. Theraulaz, J.-L. Deneubourg, A. Lioni, F. Libert, C. Sauwens, and L. Passera, "Dripping faucet with ants", *Physical Review E*, vol. 57, pp. 5904-5907, 1998.

[80] J.C. Bednarz, "Cooperative hunting in harris' hawks (Parabuteo unicinctus)", *Science*, vol. 239, pp. 1525-1527, 1988.

[81] M. Dorigo, E. Bonabeau, and G. Theraulaz, "Ant algorithms and stigmergy", *Future Generation Computer Systems*, vol. 16, pp. 851-871, 2000.

[82] G. Theraulaz and E. Bonabeau, "Modeling the collective building of complex architectures in social insects with lattice swarm", *Journal of Theoretical Biology*, vol. 177, pp. 381, 1995.

[83] R.C. Eberhart and J. Kennedy, "A New Optimizer Using Particle Swarm Theory", presented at Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, pp. 39-43, 1995.

[84] A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed Optimization by ant colonies", presented at First European Conference on Artificial Life (ECAL'91), Paris, France, pp. 134-142, 1991.

[85] E. Bonabeau, F. Henaux, S. Guerin, D. Snyers, P. Kuntz, and G. Theraulaz, *Routing in Telecommunications Networks with 'Smart' Ant-like Agents*, Intelligent Agents for Telecommunications Applications, Volume 36 in Frontiers in Artificial Intelligence and Applications, 1998.

[86] T. Arai and J. Ota, "Motion planning of multiple robots", *IEEE/RSJ IROS*, pp. 1761-1768, 1992.

[87] V.A. Cicirello and S.F. Smith, "Wasp nests for self-configurable factories", presented at Fifth International Conference on Autonomous Agents, Montreal, Canada, pp. 473-480, 2001.

[88] D.W. Van der Merwe and A.P. Engelbrecht, "Data clustering using Particle Swarm Optimisation", presented at IEEE Congress on Evolutionary Computation 2003 (CEC 2003), Canberra, Australia, pp. 215-220, 2003.

[89] L. Messerschmidt and A.P. Engelbrecht, "Learning to play games using a PSO-based competitive learning approach", presented at 4th Asia-Pacific Conference on Simulated Evolution and Learning 2002 (SEAL 2002), Singapore, pp. 444-448, 2002.

[90] B. Holldobler and E.O. Wilson, *The Ants*. Berlin: Springer, 1990.

[91] B. Holldobler and E.O. Wilson, *Journey to the Ants: A Story of Scientific Exploration*. Cambridge, USA: Harvard University Press, 1994.

[92] N.R. Franks, "Army Ants: A Collective Intelligence", *Amerian Scientist*, vol. 77, pp. 139-145, March-April 1989.

[93] S. Goss, S. Aron, J.-L. Deneubourg, and J.M. Pasteels, "Self-organised shortcuts in the Argentine ant", *Naturwissenschaften*, vol. 76, pp. 579-581, 1989.

[94] R.S. Sutton, "Reinforcement Learning Architectures for Animats", presented at From Animals to Animats:Proceedings of the First International Conference on Simulation of Adaptive Behavior, Cambridge, UK, pp. 288-296, 1990.

[95] M. Dorigo, "Optimization, Learning and Natural Algorithms (in Italian)", Ph.D. thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1992.

[96] M. Dorigo, V. Maniezzo, and A. Colorni, "An investigation of some properties of an ant algorithm", presented at Parallel Problem Solving from Nature Conference (PPSN'92), Brussels, Belgium, pp. 502-520, 1992.

[97] M. Dorigo, V. Maniezzo, and A. Colorni, "The Ant System: Optimization by a colony of cooperating agents", *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, vol. 26, pp. 29-41, 1996.

[98] L.M. Gambardella and M. Dorigo, "Solving Symmetric and Assymetric TSPs by Ant Colonies", presented at IEEE International Conference on Evolutionary Computation, pp. 622-627, 1996.

[99] J. Levine and F. Ducatelle, "Ant Colony Optimisation and Local Search for Bin Packing and Cutting Stock Problems", *Journal of the Operational Research Society, Special Issue on Local Search*, vol. 55(7), pp. 705-716, 2003.

[100] M. den Besten, T. Stutzle, and M. Dorigo, "Ant Colony Optimization for the Total Weighted Tardiness Problem", presented at 6th International Conference on Parallel Problem Solving from Nature, Paris, France, pp. 16-20, 2000.

[101] T. Stützle and M. Dorigo, "ACO Algorithms for the Quadratic Assignment Problem", in *New Ideas in Optimization*, D. Corne, M. Dorigo, and F. Glover, Eds. London: McGraw-Hill, pp. 33-50, 1999.

[102] L.M. Gambardella and M. Dorigo, "Ant-Q: A reinforcement learning approach to the traveling salesman problem", presented at Twelfth International Conference on Machine Learning (ML'95), Palo Alto, USA, pp. 252-260, 1995.

[103] M. Dorigo and L.M. Gambardella, "Ant Colonies for the Traveling Salesman Problem", *BioSystems*, vol. 43, pp. 73-81, 1997.

[104] L.M. Gambardella, E.D. Taillard, and M. Dorigo, "Ant colonies for the quadratic assignment problem", *Journal of the Operational Research Society*, vol. 50, pp. 167-176, 1999.

[105] B. Bullnheimer, R.F. Hartl, and C. Strauss, "Applying the Ant System to the vehicle routing problem", in *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, I. H. Osman, Ed. Dordrecht: Kluwer Academic Publishers, pp. 285-296, 1999.

[106] L.M. Gambardella and M. Dorigo, "HAS-SOP: A hybrid Ant System for the sequential ordering problem", *Technical Report,* IDSIA, Lugano, Switzerland, 1997.

[107] D. Costa and A. Hertz, "Ants can colour graphs", *Journal of the Operational Research Society*, vol. 48, pp. 295-305, 1997.

[108] V. Maniezzo, "Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem", *INFORMS Journal on Computing*, vol. 11, pp. 358-369, 1999.

[109] R. Michel and M. Middendorf, "An ACO algorithm for the shortest supersequence problem", *New Ideas in Optimization*, pp. 51-61, 1999.

[110] V. Maniezzo and A. Carbonaro, "An ANTS heuristic for the frequency assignment problem", *Future Generation Computer Systems*, vol. 16, pp. 927-935, 2000.

[111] H.R. Lourenco and D. Serra, "Adaptive approach heuristics for the generalized assignment problem", *Technical Report,* Universitat Pompeu Fabra, Department of Economics and Management, Barcelona, Spain, 1998.

[112] A. Bauer, B. Bullnheimer, R.F. Hartl, and C. Strauss, "An ant colony optimization approach for the single machine total tardiness problem", presented at 1999 Congress on Evolutionary Computation (CEC'99), Piscataway, USA, pp. 1445-1450, 1999.

[113] L.M. Gambardella, E.D. Taillard, and G. Agazzi, "MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows", in *New Ideas in Optimization*, F. Glover, Ed. London, UK: McGraw-Hill, pp. 63-76, 1999.

[114] G. Leguizamon and Z. Michalewicz, "A new version of Ant System for subset problems", presented at 1999 Congress on Evolutionary Computation (CEC'99), Piscataway, USA, pp. 1459-1464, 1999.

[115] Y.-C. Liang and A.E. Smith, "An Ant System approach to redundancy allocation", presented at 1999 Congress on Evolutionary Computation (CEC'99), Piscataway, USA, pp. 1478-1484, 1999.

[116] C. Solnon, "Solving permutation constraint satisfaction problems with artificial ants", presented at 14th European Conference on Artificial Intelligence, Amsterdam, The Netherlands, pp. 118-122, 2000.

[117] V.A. Cicirello and S.F. Smith, "Improved routing wasps for distributed factory control", *IJCAI-01 Workshop on Artificial Intelligence and Manufacturing: New AI Paradigms for Manufacturing*, pp. unknown, August 2001.

[118] R.S. Parpinelli, H.S. Lopes, and A.A. Freitas, "Data Mining with an Ant Colony Optimization Algorithm", *IEEE Transactions on Evolutionary Computation, special issue on Ant Colony Algorithms*, vol. 6, pp. 321-332, 2002.

[119] R.S. Parpinelli, H.S. Lopes, and A.A. Freitas, "An Ant Colony Algorithm for Classification Rule Discovery", in *Data Mining: a Heuristic Approach*, C. Newton, Ed. London: Idea Group Publishing, pp. 191-208, 2002.

[120] A. Broggi and A. Fascioli, "Artificial Vision in Extreme Environments for Snowcat Tracks Detection", *IEEE Transactions on Intelligent Transportation Systems*, vol. 3, pp. 162-172, 2002.

[121] N. Labroche, N. Monmarche, and G. Venturini, "A new clustering algorithm based on the chemical recognition system of ants", presented at 15th European Conference on Artificial Intelligence, pp. 345-349, 2002.

[122] R. Ouiddir, M. Rahli, R. Meziane, and A. Zeblah, "Ant Colony Optimization for new redesign problem of multi-state electrical power systems", *Journal of Electrical Engineering*, vol. 55, pp. 57-63, 2004.

[123] F. Fenet and C. Solnon, "Searching for Maximum Cliques with Ant Colony Optimization", presented at Applications of Evolutionary Computing: EvoWorkshops 2003, Essex, UK, pp. 236-245, 2003.

[124] P. Korosec, J. Silc, and B. Robic, "A multilevel ant-colony optimization algorithm for mesh partitioning", *International Journal of Pure and Applied Mathematics*, vol. 5, pp. 143-159, 2003.

[125] H.P. Guo, P.R. Boddhireddy, and W.H. Hsu, "An Ant Colony Optimization (ACO) Algorithm for the Most Probable Explanation Problem", presented at 17th Australian Joint Conference on Artificial Intelligence, Cairns, Australia, pp. 778-790, 2004.

[126] R. Jensen and Q. Shen, "Finding Rough Set Reducts with Ant Colony Optimization", *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, pp. 1457-1471, 2004.

[127] M. Nourelfath and N. Nahas, "Ant Colony Optimization to Redundancy Allocation for Multi-state Systems", presented at 4th International Conference on Mathematical Methods in Reliability (MMR'04), Santa Fe, USA, pp. unknown, June 2004.

[128] M.P. Oakes, "Ant Colony Optimisation for Stylometry: The Federalist Papers", presented at 5th International Conference on Recent Advances in Soft Computing (RASC2004, Nottingham, UK, pp. 16-18, December 2004.

[129] J. Green, J.L. Whalley, and C.G. Johnson, "Automatic Programming with Ant Colony Optimization", presented at 2004 UK Workshop on Computational Intelligence, Loughborough, UK, pp. 70-77, 2004.

[130] H. Chen and A.M.K. Cheng, "Applying Ant Colony Optimization to the Partitioned Scheduling Problem for Heterogeneous Multiprocessors", *ACM Special Interest Group on Embedded Systems (SIGBED) review*, vol. 2, pp. work-in-progress, April 2005.

[131] N.V. Karadimas, G. Kouzas, I. Anagnostopoulos, V. Loumos, and E. Kayafas, "Ant Colony Optimization for Municipal Services", presented at European Conference on Modelling and Simulation (ECMS), Riga, Latvia, pp. 381-386, 2005.

[132] S. Christodoulou, "Optimal Truss Design Using Ant Colony Optimization", *Accepted for publication in proceedings of Fifth GRACM International Congress on Computational Mechanics*, Limassol, Cyprus, June 2005.

[133] M.R. Jalali, A. Afshar, and M.A. Marino, "Reservoir Operation by Ant Colony Optimization Algorithms", *Iranian Journal of Science & Technology*, vol. in press, 2005.

[134] T.V. Levanova, "The Ant Colony Optimization Algorithm For Some Discrete Location Problems", presented at 12th International Conference of the European Woman in Mathematics (EWM) Association, Volvograd, Russia, pp. 18-24, September 2005.

[135] R. Garlick and R. Barr, "Dynamic wavelength routing in WDM networks via Ant Colony Optimization", presented at 3rd International Workshop on Ant Algorithms (ANTS 2002), Brussels, Belgium, pp. 250-255, September 2002.

[136] J.M. Pasteels, J.-L. Deneubourg, and S. Goss, "Self-organization Mechanisms in Ant Societies (1): Trial Recuitment in Newly Discovered Food Sources", *Experientia Suppl.*, vol. 76, pp. 579-581, 1989.

[137] M. Dorigo and G. Di Caro, "The ant colony optimization meta-heuristic", in *New Ideas in Optimization*, F. Glover, Ed. London, UK: McGraw-Hill, 1999.

[138] M. Dorigo and G. Di Caro, "AntNet: Distributed Stigmergetic Control for Communications Networks", *Journal of Artificial Intelligence Research*, vol. 9, pp. 317-365, 1998.

[139] M. Heissenbuttel and T. Braun, "Ants-Based Routing in Large Scale Mobile Ad Hoc Networks", presented at 13th ITG/GI-Fachtagung Kommunikation in verteilten Systemen (KiVS 2003), Leipzig, Germany, pp. 91-99, February 2003.

[140] T. White and B. Pagurek, "Towards Multi-Swarm Problem Solving in Networks", presented at Third International Conference on Multi-Agent Systems (ICMAS'98), pp. 333-340, 1998.

[141] T. White, B. Pagurek, and D. Deugo, "Collective Intelligence and Priority Routing in Networks", presented at 15th International Conference on Industrial and Engineering, Applications of Artificial Intelligence and Expert Systems: Developments in Applied Artificial Intelligence, pp. 790-800, 2002.

[142] S.R. Das, R.J. Marks, M.A. El-Sharkawi, P. Arabshahi, and A.A. Gray, "The Minimum Power Broadcast Problem In Wireless Networks: An Ant Colony System Approach", presented at IEEE CAS Workshop on Wireless Communications and Networking, Pasadena, USA, pp. 5-6, September 2002.

[143] G. Navarro Varela and M.C. Sinclair, "Ant colony optimisation for virtual-wavelength-path routing and wavelength allocation", presented at 1999 Congress on Evolutionary Computation, Washington, USA, pp. 1809-1816, July 1999.

[144] J.R.L. Fournier and S. Pierre, "Assigning Cells to Switches in Mobile Networks Using an Ant Colony Optimization Heuristic", *Computer Communications*, vol. 28(1), pp. 65-73, 2004.

[145] S. Rajagopalan and Chien Chung Shen, "A Routing Suite for Mobile Ad hoc Networks using Swarm Intelligence", *Technical Report,* DEGAS Networking Group, University of Delaware, Newark, USA, May 2004.

[146] J.S. Baras and H. Mehta, "A Probabilistic Emergent Routing Algorithm for Mobile Ad hoc Networks", presented at WiOpt'03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, Sophia-Antipolis, France, pp. unknown, March 2003.

[147] M. Gues, U.O. Sorges, and I. Bouazizi, "ARA - The Ant-Colony Based Routing Algorithm for MANETS", presented at 2002 International Conference on Parallel Processing Workshops, pp. 18-21, August 2002.

[148] M. Roth and S. Wicker, "Termite: Emergent Ad Hoc Networking", presented at The Second Mediterranean Workshop on Ad-Hoc Networks, pp. unknown, 2003.

[149] G. Di Caro, F. Ducatelle, and L.M. Gambardella, "AntHocNet: an Ant-Based Hybrid Routing Algorithm for Mobile Ad Hoc Networks", in *8th International Conference on Parallel Problem Solving from Nature (PPSN VIII), number 3242 in Lecture Notes in Computer Science*. Birmingham, UK, September 2004.

[150] M. Shivanajay, C.K. Tham, and D. Srinivasan, "Mobile Agents based Routing Protocol for Mobile Ad Hoc Networks", presented at IEEE GLOBECOM 2002, Symposium on Ad Hoc Wireless Networks (SAWN 2002), Taipei, Taiwan, pp. unknown, Nov 2002.

[151] M.A. El-Sharkawi, R.J. Marks, P. Arabshahi, and A.A. Gray, "Adaptive-SDR: Adaptive Swarm-based Distributed Routing", presented at 2002 International Joint Conference on Neural Networks, Honolulu, Hawaii, pp. 2878-2883, 2002.

[152] I. Kassabalidis, S.R. Das, M.A. El-Sharkawi, R.J. Marks, P. Arabshahi, and A.A. Gray, "Intelligent Routing and Bandwidth Allocation in Wireless Networks", presented at NASA Earth Science Technology Conference, College Park, USA, pp. unknown, August 2001.

[153] P. Cardoso, M. Jesus, and A. Marquez, "MONACO - Multi-Objective Network Optimisation Based on an ACO", *Technical Report,* University of Seville, Seville, June 2003.

[154] B. Awerbuch, D. Holmer, and H. Rubens, "Swarm Intelligence Routing Resilient to Byzantine Adversaries", presented at IEEE International Zurich Seminar on Communications (IZS 2004), ETH, Zurich, Switzerland, pp. unknown, February 2004.

[155] Cambridge University PressCambridge Online Dictionary, www.dictionary.cambridge.org, *accessed* 2005.

[156] A.A. Michelson, "On the application of interference methods to astronomical measurements", *Philosophical Magazine*, vol. 30, pp. 1, July 1890.

[157] J. Boleng, W. Navidi, and T. Camp, "Metrics to Enable Adaptive Protocols for Mobile Ad Hoc Networks", presented at International Conference on Wireless Networks (ICWN '02), Las Vegas, USA, pp. 293-298, 2002.

[158] C. Schurgers, V Raghunathan, and M.B. Srivastava, "Power management for energy-aware communication systems", *ACM Transactions on Embedded Computing Systems*, vol. 2, pp. 431-447, August 2003.

[159] Y. Cui, Y. Xue, and K. Nahrstedt, "Optimal Resource Allocation in Overlay Multicast", *Accepted for publication in IEEE Transactions on Parallel and Distributed Systems*, 2005.

[160] C. Tschudin, H. Lundgren, and E. Nordström, "Embedding MANETs in the Real World", presented at 8th IFIP International Conference on Personal Wireless Communications (PWC2003), Venice, Italy, pp. 578-589, September 2003.

[161] IEEE, "Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", *IEEE Standard 802.11*, 1999.

[162] National Institute of Standards and Technology (formerly National Bureau of Standards) Natrella, "Experimental Statistics: NBS Handbook 91", pp. 2-13 - 2-15, October 1966.

[163] K. Ramachandra and H.H. Ali, "Evaluating the Performance of Various Architectures for Wireless Ad Hoc Networks", presented at 37th Hawaii International Conference on System Sciences (HICSS), Hawaii, USA, pp. unknown, 2004.

[164] D. Cavin, Y. Sasson, and A. Schiper, "On the accuracy of MANET simulators", presented at ACM Workshop on Principles of Mobile Computing (POMC 2002), Toulouse, France, pp. 38-43, October 2002.

[165] I. Ari, N. Jethani, A. Rangnekar, and S. Natarajan, "Performance Analysis and Comparison of Ad-hoc Routing Protocols", *Technical Report,* Department of Computer Science and Electrical Engineering, University of Maryland, May 2000.

[166] C.P. Chu, C.P. Chang, C.W. Yeh, and Y.F. Yeh, "An On-Demand Routing Protocol with Backtracking for Mobile Ad Hoc Networks", presented at IEEE Wireless Communications and Networking Conference 2004 (WCNC 2004), pp. 1545-1550, March 2004.

[167] T. Clausen, L. Viennot, T. Olesen, and N. Larsen, "Investigating data broadcast performance in mobile ad-hoc networks", presented at Fifth International Symposium on Wireless Personal Multimedia Communications, Aalborg University and project Hipercom, INRIA Rocquencourt, pp. unknown, 2002.

[168] P. Johansson, T. Larsson, N. Hedman, and B. Mielczarek, "Routing protocols for mobile ad-hoc networks - a comparative performance analysis", presented at 5th International Conference on Mobile Computing and Networking (ACM MOBICOM'99), pp. 195-206, August 1999.

[169] Y. Lu, W.C. Wang, Y.H. Zhong, and B.K. Bhargava, "Study of Distance Vector Routing Protocols for Mobile Ad Hoc Networks", presented at First IEEE International Conference on Pervasive Computing and Communications (PerCom'03), Fort Worth, USA, pp. 23-26, March 2003.

[170] T. Stützle and H.H. Hoos, "The MAX-MIN Ant System and local search for the travling salesman problem", presented at 1997 IEEE International Conference on Evolutionary Computation (ICEC'97), Piscataway, USA, pp. 309-314, 1997.

[171] T. White, B. Pagurek, and A. Bieszczad, "Mobile agents for network management", *IEEE Communication Surveys*, vol. 1, pp. unknown, 1998.

[172] Atmel Corporation Website, www.atmel.com, *accessed* 2005.

[173] Koninklijke Philips Electronics N.V. Website, www.semiconductors.philips.com, *accessed* 2005.

[174] S.J. Lee, E.M. Belding-Royer, and C. Perkins, "Scalability study of the ad hoc on-demand distance vector routing protocol", *International Journal of Network Management*, vol. 13, pp. 97-114, 2003.

[175] R. Castaneda and S.R. Das, "Query Localization Techniques for On-demand Routing Protocols in Ad Hoc Networks", presented at ACM/IEEE International Conference on Mobile Computing and Networking, Seattle, USA, pp. 186-194, August 1999.

[176] V. Raghunathan, C. Schurgers, S. Park, and M.B. Srivastava, "Energy-Aware Wireless Microsensor Networks", *IEEE Signal Processing Magazine*, vol. 19(2), pp. 40-50, March 2002.

Appendix A: Ad Hoc Routing Protocols

ABR	Associativity Based Routing
ACODV	Ant-Colony Optimisation Distance Vector Routing
ADV	Adaptive Distance Vector Routing
AODV	Ad Hoc On Demand Distance Vector Routing
BRP	Bordercast Resolution Protocol
CBRP	Cluster Based Routing Protocol
CEDAR	Core-Extraction Distributed Ad Hoc Routing
CGSR	Clusterhead Gateway Switch Routing
DDR	Distributed Dynamic Routing
DREAM	Distance Routing Effect Algorithm for Mobility
DSDV	Destination Sequenced Distance Vector Routing
DSR	Dynamic Source Routing
DST	Distributed Spanning Tree
FORP	Flow Oriented Routing Protocol
FSLS	Fuzzy Sighted Link State
FSR	Fisheye State Routing
GEDIR	Geographic Distance Routing
GPSR	Greedy Perimeter Stateless Routing
GSR	Global State Routing
HSR	Hierarchical State Routing
IMEP	Internet MANET Encapsulation Protocol
LANMAR	Landmark Routing Protocol
LAR	Location Aided Routing
LMR	Lightweight Mobile Routing
LRR	Link Reversal Routing
LUNAR	Lightweight Underlay Network Ad hoc Routing
MMBDP	Mobile Mesh Border Discovery Protocol
MMLDP	Mobile Mesh Link Discovery Protocol
MMRP	Mobile Mesh Routing Protocol
MMWN	Multimedia Support in Wireless Networks
OLSR	Optimized Link State Routing
RDMAR	Relative Distance Micro-discovery Ad Hoc Routing

ROAM	Routing on-demand acyclic multi-path
SLURP	Scalable Location Update Routing Protocol
SSA	Signal Stability-Based Adaptive Routing
STAR	Source Tree Adaptive Routing
TBRPF	Topology Broadcast Based on Reverse Path Forwarding
TLR/TRR/AGPF	Terminode Routing
TORA	Temporally Ordered Routing Algorithm
WAR	Witness Aided Routing
WRP	Wireless Routing Protocol
ZHLS	Zone-based Hierarchical Link State
ZRP	Zone Routing Protocol

Appendix B: Definition of symbols

$\tau_{kn}{}^{d}$	The pheromone level at node *k* at time *t* associated with using node *n* as a next-hop destination for an ant traveling to node *d*.
$p_n{}^{d}$	Probability value $p_n{}^{d}$ which indicates the "attractiveness" of that pair,
$U(0,1)$	A uniformly distributed random number, $1 \geq U \geq 0$.
k_r	Tunable system parameter that determine the amount of pheromone contributed by the random term *U*.
k_l	Tunable system parameter that determine the amount of pheromone contributed by the route length D_{ko}.
ε	Pheromone update interval
τ_{max}	Maximum pheromone level
τ_{min}	Minimum pheromone level
p_{dard}	The probability than a node receiving a data packet as final destination will launch a RREQ packet with the SendRREP bit set to false (i.e., a seed packet)
ρ	Pheromone evaporation rate, $0 \leq \rho \leq 1$.
α	Pheromone amplification factor