Idris Skloul Ibrahim, PhD Student

Telephone:	+44 (0) 78 288 14391 +44 (0)131 451 8283
Fax:	+44 (0)131 451 3327
Email:	skloul@garyounis.edu isi3@hw.ac.uk
Location:	Mountbatten, Room G28

My PhD Proposal:

PhD Research Proposal

An optimum routing Algorithm for Ad Hoc Networks

By

Idress Skloul Ibrahim

Supervisor: Dr. Peter King

Introduction

In recent years, mobile computing has enjoyed a tremendous rise in popularity. The continued minimization of mobile computing devices and the extraordinary rise of processing power available in mobile laptop computers combine to put more and better computer-based applications into the hands of a growing segment of the population. At the same time, the markets for wireless telephones and communication devices are experiencing rapid growth. Projections have been made that, in nowadays there are more than billion wireless devices in use. Therefore, the wireless mobile computers or *Mobile Ad Hoc Networks* (MANET) have become very necessary.

A wireless ad hoc network is a collection of autonomous nodes or terminals that communicate with each other by forming a multihop radio network and maintaining connectivity in a decentralized manner. Since the nodes communicate over wireless links, they have to contend with the effects of radio communication, such as noise, fading, and interference. In addition, the links typically have less bandwidth than in a wired network. Each node in a wireless ad hoc network functions as both a host and a router, and the control of the network is distributed among the nodes.

The network topology is in general dynamic, because the connectivity among the nodes may vary with time due to node departures, new node arrivals, and the possibility of having mobile nodes. Hence, there is a need for efficient routing protocols to allow the nodes to communicate over multihop paths consisting of possibly several links in a way that does not use any more of the network "resources" than necessary. Some of these features are characteristic of the type of packet radio networks that were studied extensively in the 1970s and 1980s. To provide routes in such dynamic environments, many routing protocols have been proposed over the last few years. These protocols can be broadly classified onto three categories, namely, ***proactive***, ***reactive***, and ***hybrid***.

Yet, research in the area of ad hoc networking is receiving much attention from academia, industry, and government. Since these networks pose many complex issues, there are many open problems for research and opportunities for making significant contributions, one of these issues is the communication cost during the route discovery. This proposal is concerns with the development and evaluation of **O**ptimum **R**outes **(OR**s**)** discovery between a source node and a destination in the MANET.

Wireless Data Rates

One of the biggest obstacles to the adoption of ad hoc networks may be reduced data rates, the same problem that slowed the adoption of wireless computing during the last decade. We can typically observe an order of magnitude difference in the speed of wired and wireless networks. For instance, while many enterprise users are accustomed to 100 Mbit/sec from the local Ethernet, wireless users must struggle to get a reliable 10 Mbit/sec over the air: 1 to 2 Mbit/sec is much more common. The overall effect tends to be that wireless computers are no longer general purpose. The wireless user has to be careful not to invoke applications that require a lot of bandwidth.

The Research Proposal

This proposal studies ad hoc routing and its related issues, namely congestion control and communication cost. In the investigation of these issues, we pay specific attention to the scalability of algorithms in respect to the network size and nodal mobility. Extensive nodal mobility of MANETs makes multi-hop routing a genuine challenge. The frequent topology changes and variable propagation conditions make a routing table obsolete very quickly, which results in enormous control overhead for route discovery and maintenance. However, MANETs suffer from severe constraints on communication resources (bandwidth, radio propagation, energy supply, etc.). In some scenarios, the routing maintenance itself may consume so much in the way of resources that no bandwidth might remain for the transmission of data packets. Even worse, the short lifetime of routing information means that a portion of the information may never be useful any more, and thus the bandwidth used to distribute the routing update information could be wasted.

Proposal Aims

- Studding and investigating the current protocols of ad hoc wireless networks.
- Assessing the difficulties faced by other routing protocols.
- Designing algorithms needed to implement the idea of this proposal (The **O**ptimum **R**outing **A**lgorithm.), and achieve the following goals:
    - Less congestion control, overhead control and communication cost.
    - High speed during route discovery, update and data maintenance.
    - Packets transfer guarantee and insurance.
    - Optimum bandwidth for data packet transmission.
- Navigating features of software tools that will use in implementation, testing and comprising, such as (C++, Ns2 simulator, TCL and TK).

Proposal Questions

In order to unlock this dilemma (*Ad Hoc routing and its related issues*), one of the solutions is to make the best use of *local* information to update routing tables, avoiding the dissemination of routing messages at the global network scale.

The collective behavior of ant foraging presents the property of self-organization, namely, the optimal solutions at the global level may emerge from the local interaction of individuals that exhibit simple behavior. This property provides a heuristic to efficiently and scalable maintain the routing information of the ever-changing topology with minimum resources. In this proposal, one of the generic questions we ask is: how can the mechanism of the ant colony are used to maximize (simultaneously) the efficiency, scalability, and stability of such a system? That is, how this mechanism can help in routing using local information and to what extent the use of local information can improve the scalability of the routing algorithm.

What is the routing algorithm?

In internetworking, it is the process of moving a packet of data from source to destination. Routing is usually performed by a dedicated device called a router. Routing is a key feature of the Internet because it enables messages to pass from one computer to another and eventually reach the target machine. Each intermediary computer performs routing by passing along the message to the next computer. Part of this process involves analyzing a *routing table* to determine the best path.

Routing is often confused with *bridging,* which performs a similar function. The principal difference between the two is that bridging occurs at a lower level and is therefore more of a hardware function whereas routing occurs at a higher level where the software component is more important. And because routing occurs at a higher level, it can perform more complex analysis to determine the optimal route (*path*) for the packet.

What is the idea behind the **O**ptimum **R**oute **P**rotocol (ORP), in this proposal?

- It is a Proactive (*table driven*) for all one and two hop neighbours, and reactive (*On demand*) for all other nodes in the network.
- Using Anycast, Unicast and Multicast propagation techniques.
- Creates the following routing tables in each node, Network table (NT), (to *save the ID of each node and its radius metric in the network*), optimum routes table (ORT), (*to save all optimum routes to each needed node in the network*), and non optimum routes table (NORT), (*to save at maximum 2 routes to each needed node in the network*).
- Route repair and maintenance, carryout by the node which discovers the route failure (Local path repair) to save time and data transmission.
- Each node has more than 2 hops from the new node, no need to send any feedback to that new node.
- The new node gets information for all other nodes which have more than 2 hops, from its closest neighbour, without need to build and save routes to that nodes, it will initiate a route discovery when they needed (*On demand*).
- Each node in the network should gets information from any route pass through it.

What is the idea behind the **O**ptimum **R**oute **A**lgorithm (ORA)?

The optimum route (*OR*) is that route between a source node and a destination, providing the following specifications:

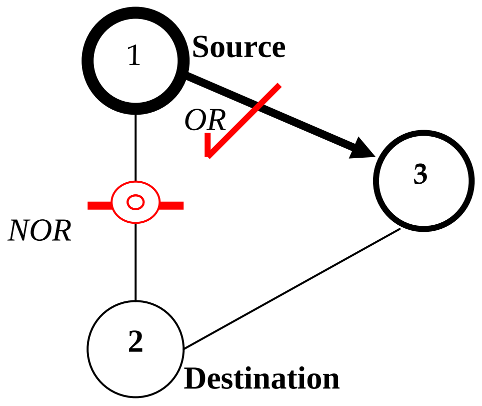
- It is the only route, if the source and the destination are one hop neighbours.
- It is the shortest route between the source and destination.
- Doesn’t cross with any other route through any one of the source’s neighbours to the same destination, unless the joint node is one of the Destination’s neighbours. (*No joint node (JN) or joint link (JK)*).

How many Optimum route (OR) between each source and destination?

Number of optimum routes depends on number of neighbours for both source and destination as shown in the following examples.

Example1:

Nodes 1, 2 and 3 are neighbours, node 1, the source node has 2 routes to the destination node 3, the first route is (1- 3), and the second route is (1-2-3) through nod 2, but the optimum route (*OR*) is only one, the direct route (1-3), because it is the shortest and the direct route, and if node 2 is moved or there is any issue within, this route remains effective while the source and the destination nodes are exist.



✓ *OR = Optimum Route* ⊝ *NOR = Non Optimum Route*

Fig (1) optimum and no route between 3 neighbours

Example2:

Nodes 1, 2, 3 and 4 are four neighbours, normally node 1, the source node has 3 routes to node 2, the destination node, the first is (1-2) the direct route, the second route is (1-3-2) through nod 3, and the third route is (1-4-2) through node 4, but because the source and the destination are one hop neighbours, there is only one optimum route *OR*, the direct and shortest route (1-2), and also because if node 3 or 4 or both are moved or have any issues, this route remains effective while the source and destination are in the network, but if there is a problem in node 1 or 2 no benefit from any other routes such like 1-3-2 or 1-4-2, by the way, these two non optimum routes (*NOR*) will not only discarded from the source's routing table (ORT), but they will save in its non optimum routing table (NORT), just if they needed, in case if there is any problem faced in the optimum route(OR).

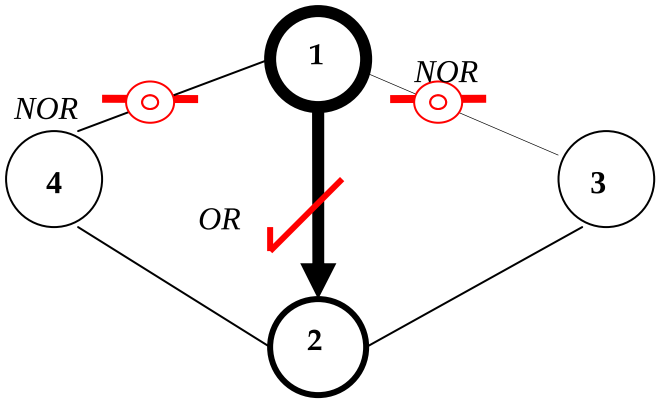


Fig (2) optimum route between 4 neighbours

Example3:

In the following network, Node 1 normally has 15 routes to Node 9, through its 3 neighbours 2,3 and 4, but node 2 is a joint node (*JN*) with node 4 *(all its routes to the destination pass through node 4*),therefore, there are only 2 optimum routes as described in the table below. The first optimum route (*OR*) uses node 4 as a second hop, while the second optimum route (*OR*), uses node 3 as a second hop to the same destination.

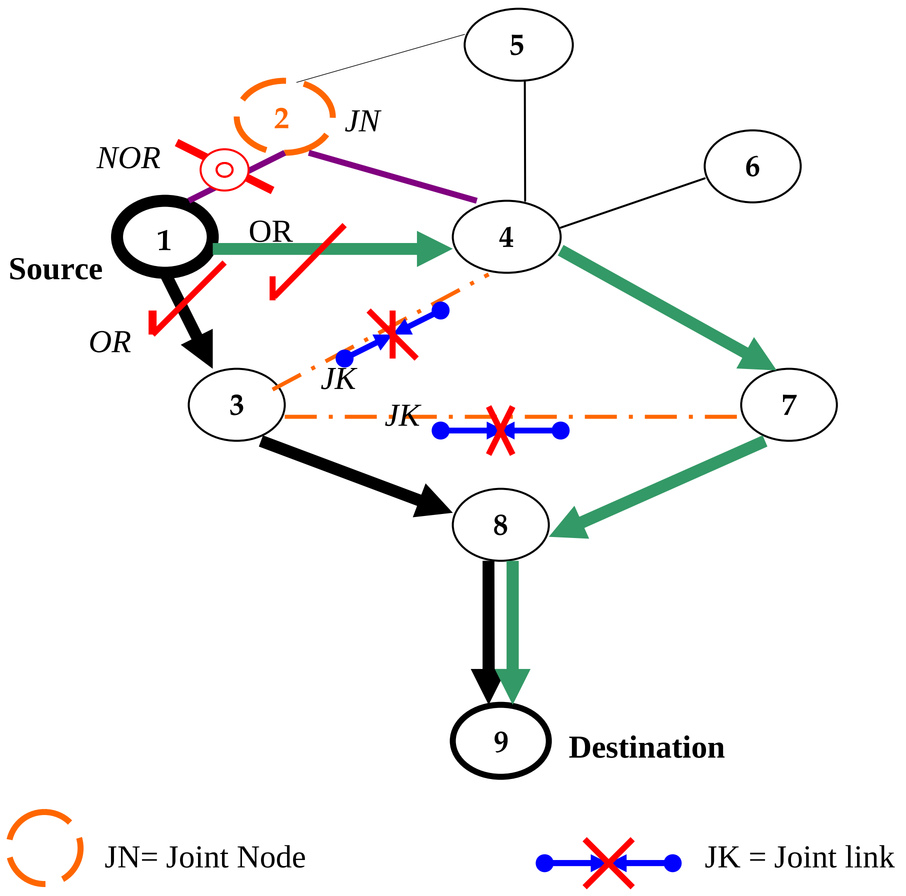


Fig (3) joint node, joint link and Optimum routes

No.	Route sequence	No. of hops	OR & NOR	Notes
1	**1-3-8-9**	**3**	√	**Shortest and optimum route**
2	**1-4-7-8-9**	**4**	√	**Short and optimum**
3	1-3-7-8-9	4	-	Joint node(3) with the shortest routes (1)
4	1-3-4-7-8-9	5	×	Joint nodes(3,4) with the shorter routes (1,2)
5	1-4-7-3-8-9	5	-	Long route, joint node(3) with shorter routes (1)
6	1-4-3-7-8-9	5	×	Long route, joint node(3) with shorter routes (1)
7	1-2-4-3-8-9	5	×	Long route, joint node(4) with shorter routes (2)
8	1-3-4-7-8-9	5	×	Long route, joint node(3) with shorter routes (1)
9	1-2-4-7-8-9	5	×	Long route, joint node(4) with shorter routes (2)
10	1-2-4-3-7-8-9	6	×	Long route, joint node(4) with shorter routes (2)
11	1-2-4-7-3-8-9	6	×	Long route, joint node(4) with shorter routes (2)
12	1-2-5-4-3-8-9	6	×	Long route, joint node(4) with shorter routes (2)
13	1-2-5-4-7-8-9	6	×	Long route, joint node(4) with shorter routes (2)
14	1-2-5-4-3-7-8-9	7	×	Long route, joint node(4) with shorter routes (12
15	1-2-5-4-7-3-8-9	7	×	Long route, joint node(4) with shorter routes (2)

√ **OR** saved in ORT   **- Nor** saved in NORT   × long and non optimum route, no need to save them

Table (1) optimum and non optimum routes

Example4:

Node 5 normally has 11 routes to Node 9, but because it has only 2 neighbours2,4 there are only 2 optimum routes as described in the table below.

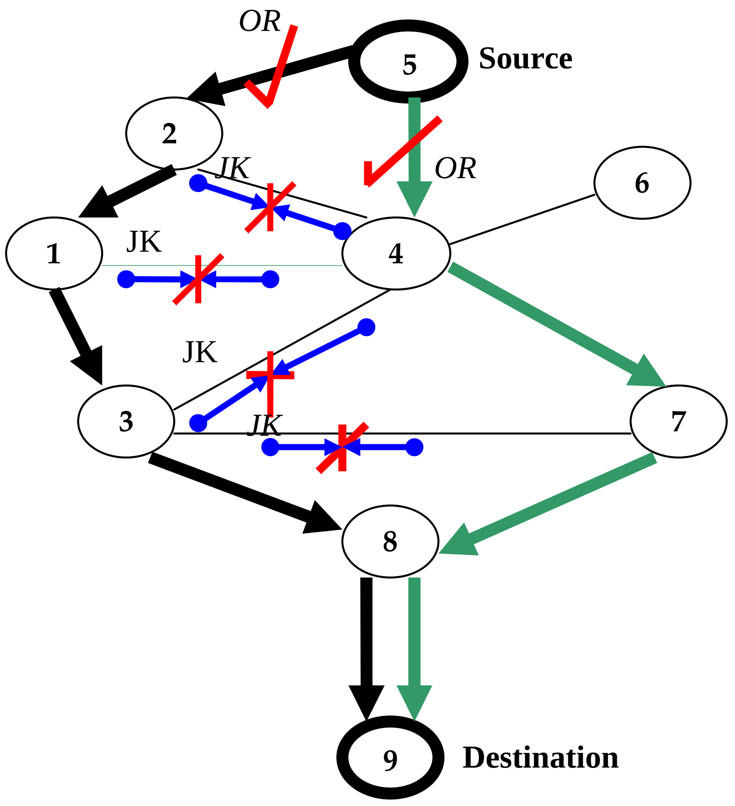


Fig (4) joint node, joint link and Optimum routes

No.	Route sequence	No. of hops	OR & NOR	Notes
1	5-4-7-8-9	4	√	Shortest and optimum
2	5-4-3-8-9	4	-	Joint node(4), (3) with other optimum routes (1), (3)
3	5-2-1-3-8-9	5	√	Short and optimum no joint route or node
4	5-4-7-3-8-9	5	×	Joint node(4), (3) with other optimum routes (1), (3)
5	5-4-3-7-8-9	5	×	Joint node(4), (3) with other optimum routes (1), (3)
6	5-2-4-7-8-9	5	-	Joint node(4) with an other optimum route (1)
7	5-2-4-3-8-9	5	×	Joint node(4), (3) with other optimum routes (1), (3)
8	5-2-4-7-3-8-9	6	×	Long and Joint node(4), with an other optimum route (1)
9	5-2-1-3-7-8-9	6	×	Long and Joint node(3), with an other optimum route (3)
10	5-2-4-3-7-8-9	6	×	Long and Joint node(4), (3) with other optimum routes (1), (3)
11	5-2-1-3-4-7-8-9	7	×	The longest route, Joint node(3), (4) with other optimum routes (1), (3) , joint link as well

√ **OR** saved in ORT   - **Nor** saved in NORT   × long and non optimum route, no need to save them

Table (2) optimum and non optimum routes

Note: All the optimum routes are save in the optimum routing table(ORT), while only the shortest non optimum routes, at maximum 2 routes to each destination are save in the non optimum routing table(NORT).Other routes are discarded without need to save them.

How the new node creates its own routing tables?

As mentioned in the research questions section, each node creates some tables using the protocol algorithms, here are the most important tables and how the new node creates its own tables, but first of all, let us have a look to some important points which need to know in this proposal.

- **Node's radius and its one hop and two hops neighbours:**

This example shows a new node, node No. 6 and its zone when joined the network. Each node has only one hop to the new node we call it a one hop neighbour where radius R=1, such as node 5 and node 4, while any node has 2 hops to the new node, we call it two hops neighbour and R=2, (i.e. node 2 and node 3).

All rest nodes which have more than 2 hops, such like node 1, we call they out of zone nodes where R>2. The radius (R) is not a physical metric it is just a number of hops from or to the new node.

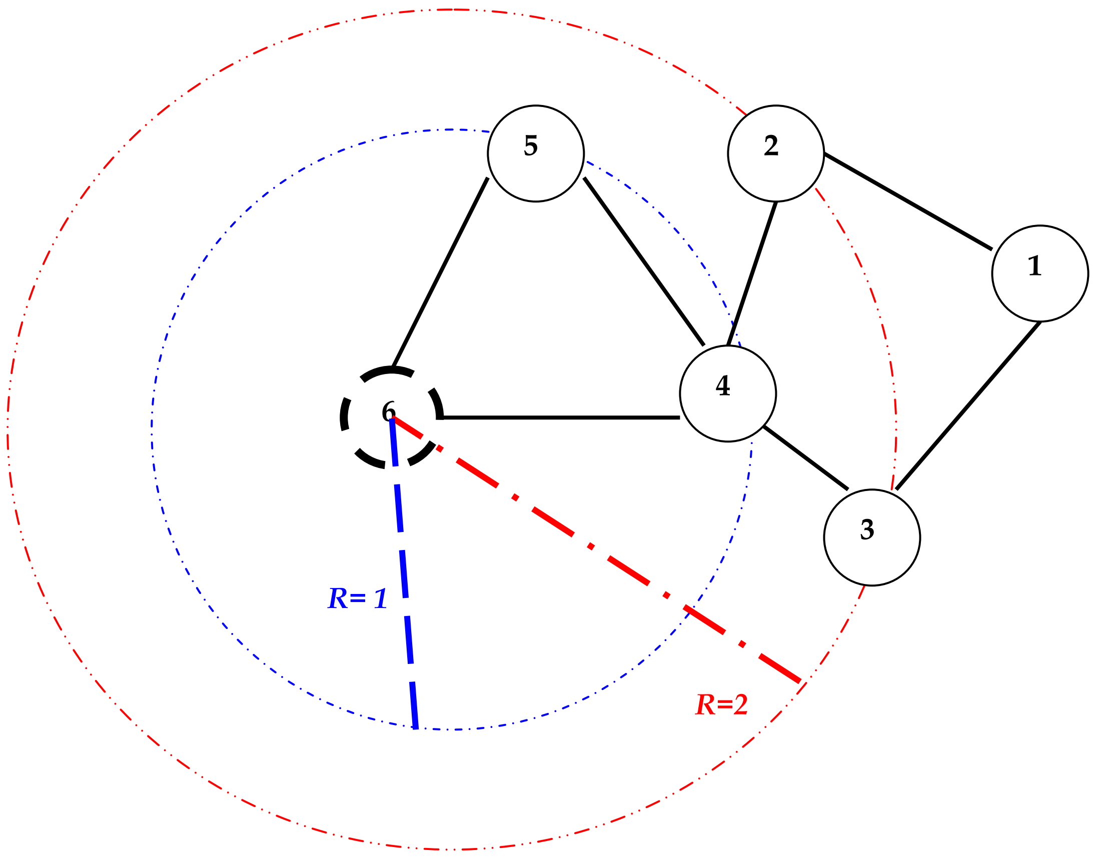


Fig (5) a new node and its zone where Radius less than or equal 2 hops

The optimum routes to all nodes where R=1 and R=2 are save in the new node's *optimum routing table* (ORT), while at maximum 2 of non optimum routes to the same nodes save in *non optimum routing table* (NORT).

The rest of the nodes where R>2 are save in the new node's *network table* (NT), with no need to find or calculate the routes (*paths*) which lead to these nodes, the routes will request when they needed on demand.

- **Optimum route algorithm's** (ORA) **messages**

  - ***Hello*** message, e*very 1 sec, sends by each an old node to confirm that, it is still in the network, and the replay should be* **Hello** *as well from each node receives it.*

  - ***Hello*** message, *no specific time, sends by each new node joined the network, and the replay should be a welcome message in this case, to differentiate between a new node joined the network and an old node still in the network.*

  - ***Welcome*** message, *is a replay message from each node recognises there is a new node joined the network where the metric R=1 or R=2.*

  - ***New node*** *message, it is a forward announcement message between the neighbours to identify there is a new node joined the network. If the metric associated with this message R<2 no replay needed, but if R=1 or 2 a welcome message should be send back to the new node through the neighbour who send the New node message.*

  - ***Route reques****t, it is a message from a node to its neighbours, when requests a route to a node out of its zone where metric R>2, any node knows the required node should replay, unless the node required is replayed itself.*

  - ***Route replay****, it is a replay message for a route request, when a required node is found by any other node or by itself.*

  - ***NT request****, it is a message from a new node to its closest one hop neighbour where R=1, to send it its whole Network table (NT), to create its won NT for future on demand route request to any node outside of its zone where R<2.*

  - ***Route Error RERR****, this is a forward message, generates by any node discovers a link break, to all its neighbours. Each node receives this message should deletes any related nodes from its NT or any related route in its ORT or NORT.*

  - ***Route Update,*** *is a forward message from the node discovered the broken link to the source node which still using the node caused the break as an intermediate node and doesn't know that node is moved.*

Methodology and Algorithms:
-In this method, we pay attention to the cost of communication regarding to the time in both send, receive and during route discovery as well. In the following section we will explain the methodology of routing protocol.

1- When a new node joined the network.
-When a new node, node 6 for example joined the network, it sends ***Hello*** message using anycast technique *where the communication takes place over a network between a single sender and all nearest nodes* (*the new node's one hop neighbours where Radius R=, node 5, 4 in this example*).

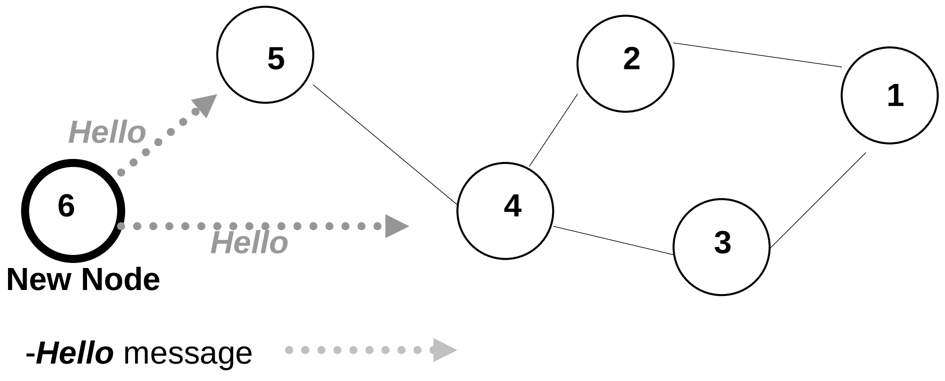


Fig (6) a new node's ***Hello*** message and its one hop neighbours

- For instant of time the new node (6) waits a ***welcome*** message from the nodes received its hello message (*its one hop neighbours* 5, *4*).

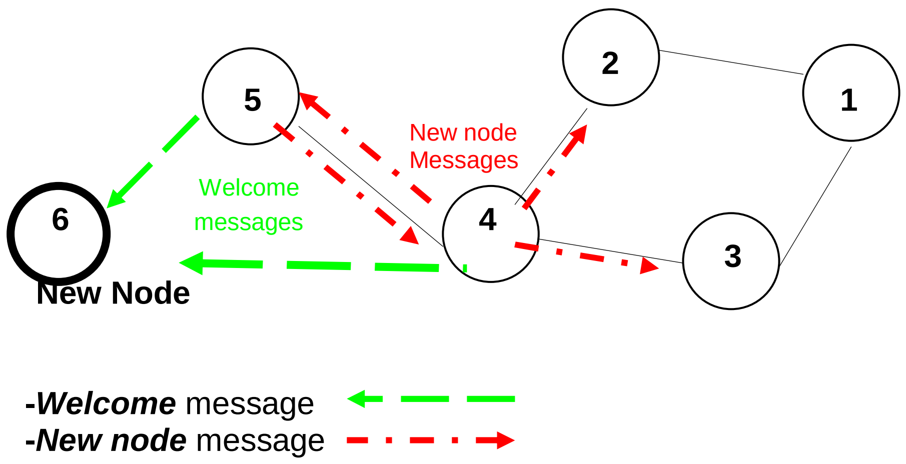


Fig (7) new node's one hop neighbours and ***welcome*** message and ***new node*** message

-Any node receives the new node's ***Hello*** message such as nodes 5 and 4, should saves the new node direct route into its optimum routing table(ORT), and announces that to its neighbours(*new node's 2 hops neighbours*) by anycasting a ***new node*** message where No. of hops=1.

-Afterwards, every node received a ***new node*** message from its neighbour, sends a ***welcome*** message as a feedback to the new node through that neighbour, except the nodes which have more than 2 hops, such as node 1 in this example, no need to send a feedback to the new node because it is outside of the new node's zone, but it should saves the new node route and associated information in its ORT and tells its neighbours about the new node by the same procedure, providing increasing No. of hops by 1 each time the message forwarded.

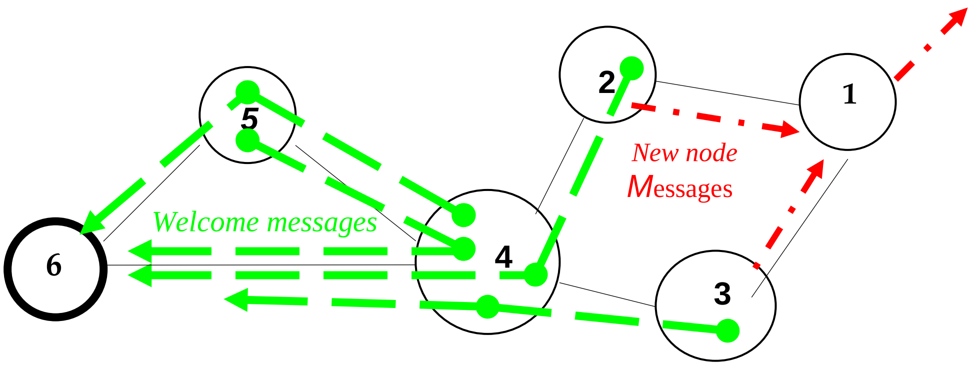


Fig (8) new node's one hop and 2 hops neighbours and their ***welcome*** messages and ***new node*** messages

-Now, every node in the network knows the new node, and has selected at least a one optimum route to it, and saved that route in its optimum routing table (ORT).

-The welcome messages and the associated information, which received by the new node from node 5, 4, 3 and 2, provided the new node all the information needed about each node in its zone where R=1 and R=2 (the *new node's one hop and two hops neighbours*), the new node should saves the shortest routes to these nodes in its ORT. Where at maximum 2 of shortest long routes save in its NORT.

-The new node selects the closest one hope neighbour and asks it to send it its NT table, to create its own network table (NT)……………

-Now, the new node knew all the nodes in the whole network, also it has already routes to any node where R=1 and R=2 in its ROT and NROT, while to any node where R>2 it has to initiate a route request, when it is needed, we will talk about rout request algorithm in the algorithms section. Let us see the new node's tables, after receiving all welcome messages.

D. ID	Route Request
1	1
2	2
3	2
4	2
5	2

Route request: 1=Reactive: 2= Proactive

Table (3) Network Table (NT)

S. ID	D. ID	Next hop	Metric (R)	Sequence Number	TTL
6	2	4	2		
6	3	4	2		
6	4	4	1		
6	5	5	1		

Table (4) Optimum routing table (ORT)

Source ID	Destination ID	Next hop	Metric (R)	Sequence Number	TTL
6	2	5	3		
6	3	5	3		
6	4	5	2		
6	5	4	2		

Table (4) Non optimum Routing table (NORT)

2- Node Movement

Mobile nodes cause broken links as they move from place to place. Any node discovers a broken link should broadcast that to its neighbours to update their tables regarding to that node which caused the broken link. In this section we are going to focus on the way by which the nodes follow to update their tables when they receive route error.

In this example, we assume that node 6 has moved away of its zone, and joined node 4's zone and node 3. Node 5 discovered and broadcasted to its neighbours that node 6 is no more reachable in its old zone by sending *RERR* message after deleting it from its tables.

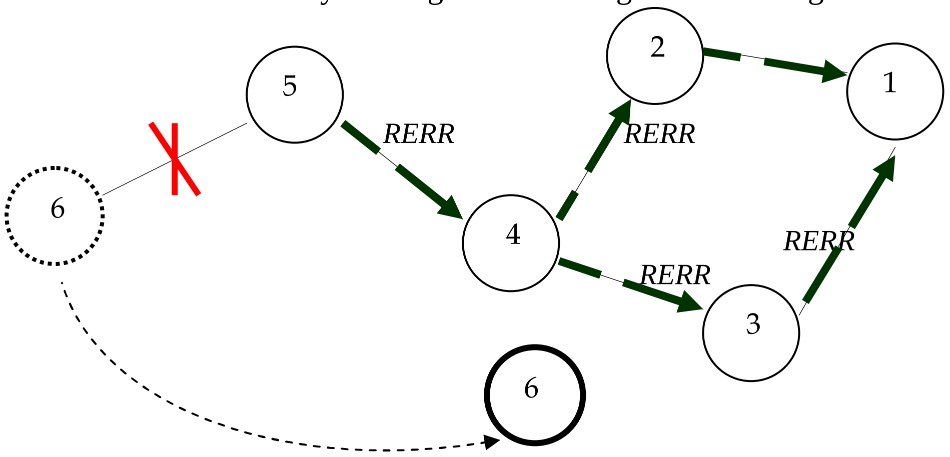


Fig (9) broken link and *RERR* message

Node 4 is the only neighbour of node 5 in this example, when it receives this message *RERR* from node 5, deletes node 6 from its NT and all related routes from its ORT and NORT, and then sends the same message (*RERR)* forward to tell its neighbours (*node 2 and 3*) that the link to node 6 is broke. Node 2 and 3 follow the same procedure, to delete node 6 from their tables and tell their neighbours to follow the same steps by forwarding the same message (*RERR*), and so on, till all nodes in the network knew that node 6 is no longer reached at its previous place and delete it from their tables.

Note: *Each node receives RERR and it is a neighbour to the node caused that Error should send hello message to that node to make sure it is not reachable, before deletes it from its tables.*

Once node 6 reaches its new place, it will join the network as a new node, and follows the same procedure described in previous section when a new node joins the network. The next figure shows graphically what happening when node 6 reached the new place.

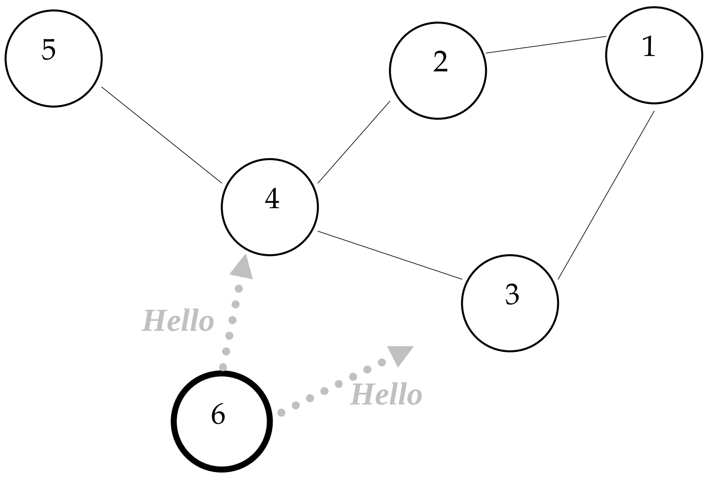


Fig (10) shows the ***Hello*** message from node 6 to its new neighbours

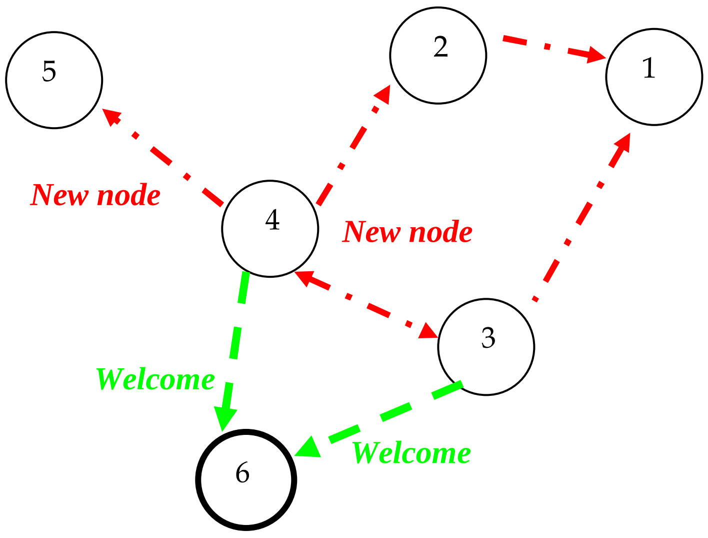


Fig (11) shows the ***Welcome*** messages to node 6 and ***new node*** messages from the new node's neighbours

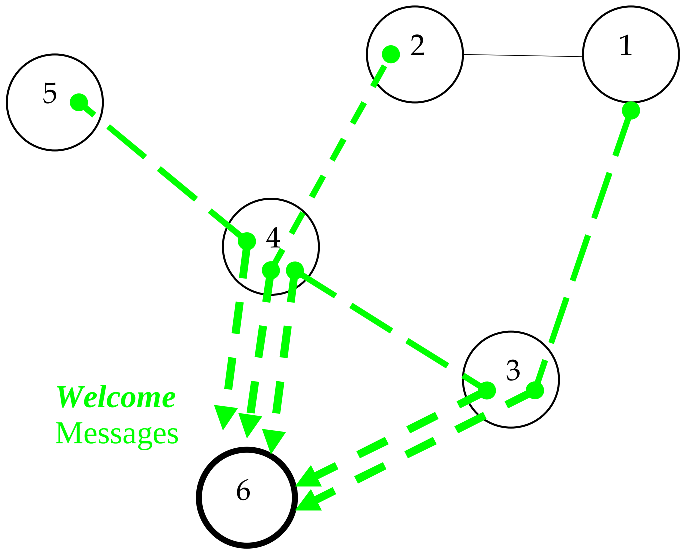


Fig (12) shows the ***Welcome*** messages from the neighbours where R=1 or R=2

Let us see the node 6's tables, after changing its place

D. ID	Route Request
1	2
2	2
3	2
4	2
5	2

Route request: 1=Reactive: 2= Proactive

Table (5) Node 6's Network Table.

S. ID	D. ID	Next hop	Metric (R)	Sequence Number	TTL
6	1	3	2		
6	2	4	2		
6	3	3	1		
6	4	4	1		
6	5	4	2		

Table (6) Node 6's Optimum routing table.

Source ID	Destination ID	Next hop	Metric (R)	Sequence Number	TTL
6	1	4	3		
6	2	5	3		
6	3	5	3		
6	4	5	2		
6	5	4	2		

Table (7) Non optimum Routing table.

3- Route maintenance

The broken link may be detected by the layer -2 protocol (*Data link layer*), or it may be inferred if no broadcasts have been received for a while from a former neighbour, a broken link is detected by a metric of ∞ (*any value greater than the maximum allowed metric*). When a link to a next hop has broken, any route through that next hop is immediately assigned infinity metric, and an updated sequence number.

Any node discovers a broken link should broadcast that to its neighbours to update their tables Once a route has discovered for a given source and destination, it is maintained as long as needed by the source node. Movement of nodes within the ad hoc network affects only the routes containing those nodes, such a path is called an *active path*, in this proposal such these paths stored in ORT or in NORT. Movement not along an active path does not trigger any action. If the source node moves during an active session, it can reinitiate route discovery to establish a new route to the destination if it hasn't another route in its ORT or in its NORT. When either the destination or some intermediate node moves, however, a Route Error (RERR) message is sent to the effected source nodes. This RERR is initiated by the node upstream of the break (the closest node), which discovered the break.

However, when a link is broken due to movement, any passive-redirector that can be found at that moment may at best repair the broken path. In other words, the path-shortening phenomenon does not always happen, and the consequence is a repaired path with increased length, or equivalently, a path with longer end-to-end delay, for any data packets that follow.

In the following example figure (13), the original path from the source node 5, to the destination node 8 is through node 4, 3 and node 10. Node 10 then moves to a new location (x, y), causing a break in connectivity with node 3. Node 3 notices this break and sends a *RERR* to all its neighbours 1, 4, 6, and Node 7.

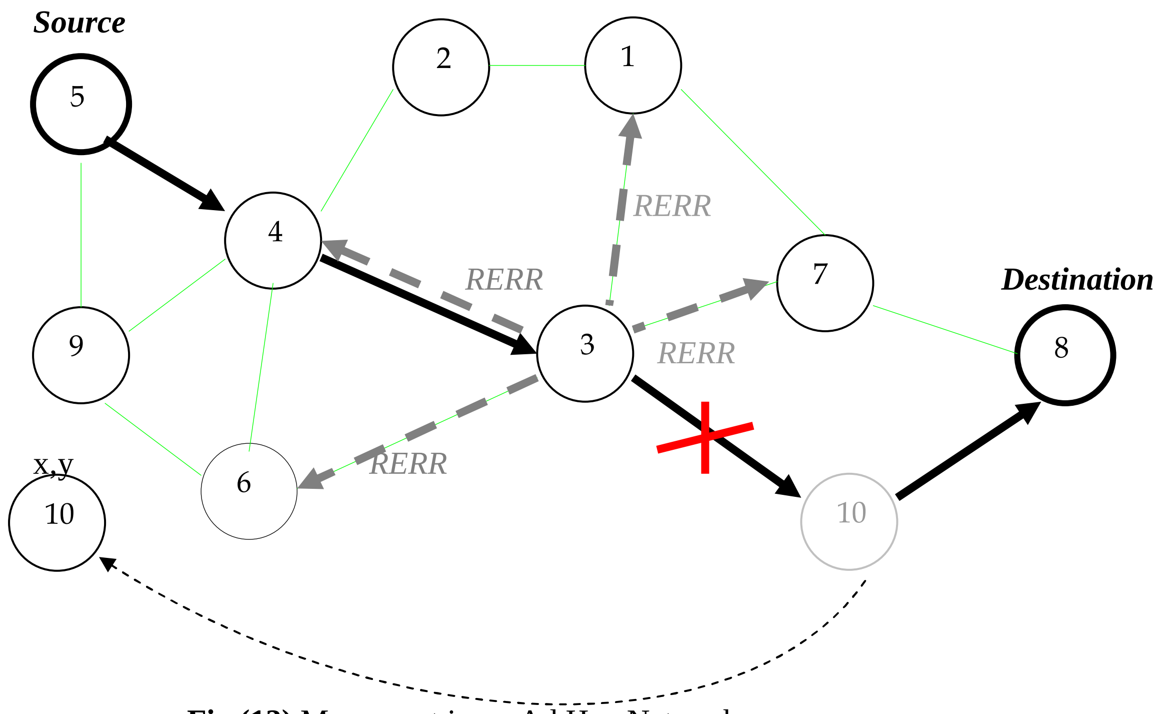


Fig (13) Movement in an Ad Hoc Network

Each node marks this route as invalid and then forwards the *RERR* to its neighbours, except the node which has a route contains node 10 (*the node which caused the break*) as a next hope, it should seek to find an other node to substitutes node 10 as a next hop (*could be there are some nodes still using node 10 as an intermediate node, and they have no idea its moved. Nodes recognise the destination during broken link and RERR message receiving, but not the intermediate node*).

In this example, only node 3 and 6 have node 10 as a next hop to node 8, therefore both should maintain their routes to the destinations where node 10 is the next hop, this is to save any packet already sent.

Figure (14) illustrates route maintenance procedure, where node 3 and node 6 immediately start to find a new route to node 8. Node 3 found a new route through node 7 and node 6 found the same route through node 3 as well.

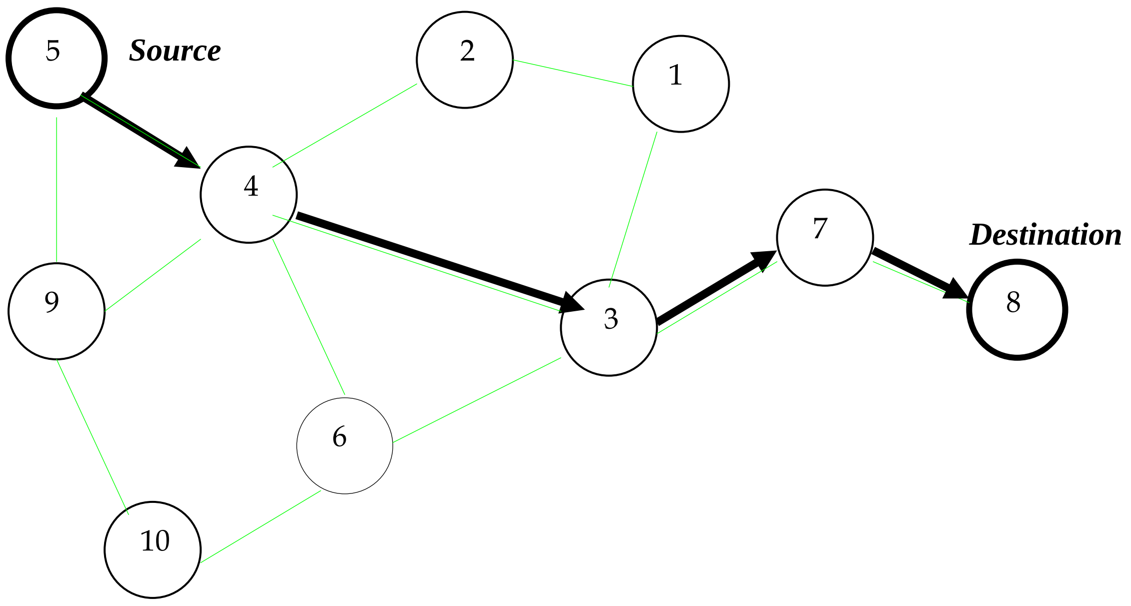


Fig (14) Responding to topology changes and route maintenance

This procedure done to save the packets sent from node the source to the destination to make sure that not lost at node 3, and to save the time till the source node determines if it still needs that route or not, and so, it can reinitiates route discovery if still needs that route to the destination.

Route maintenance procedure:

- Any node discovers the broken link, notices this break and sends *RERR* as a forward message to all its neighbours.
- All the nodes mark that route as invalid with an ∞ metric to that node.
- Route entries with an ∞ metric are not immediately deleted because they contain useful routing information.
- The node discovered the break, should repair the active route to save both the time and transferred packets between the source and the destination.
- All the nodes which have the node caused the break as a second hop to a specific destination, should find a substitute node as a next hop to that destination.

Route update

Sometimes there is a node has a route to a specific destination, and the node caused the broken link was a one of its intermediate nodes, such as node 2 for example, it has the following route to node 8 (2-4-3-10-8), because we don't save the whole route hops to the destination, but only the next hop and number of hops. Therefore, the source node doesn't know that the node caused the broken link is one of its intermediate nodes, otherwise, it will delete it as soon as received *RERR*, in this case the route update is needed and should initiated by the node which discovered the broke link.

Once node 2 used this route and sends a packet to node 8 through its next one hope neighbour (*node 4*). Node 4 has the same route to node 8 in 3 hops (4-3-10-8) through its on hop neighbour node 3, and doesn't know that node 10 is one of its intermediate nodes as well. Therefore, it forwards that packet to the destination through its next one hop neighbour (*node3*).

Finally the packet from node 2 (*source node*) to node 8 the (*destination node)* reached node 3. It is the node that discovered the broken link, and knows that the route (3-10-8) is no more useful, because of broken link between it and node 10, therefore, it forwards this packet to the destination using the new route (3-7-8), and sends a *RUPD message* to the source node to update its route to the destination node through node 4. Node 4 will get an idea that the route to node 8 is changed and should update its route to node 8 as well.

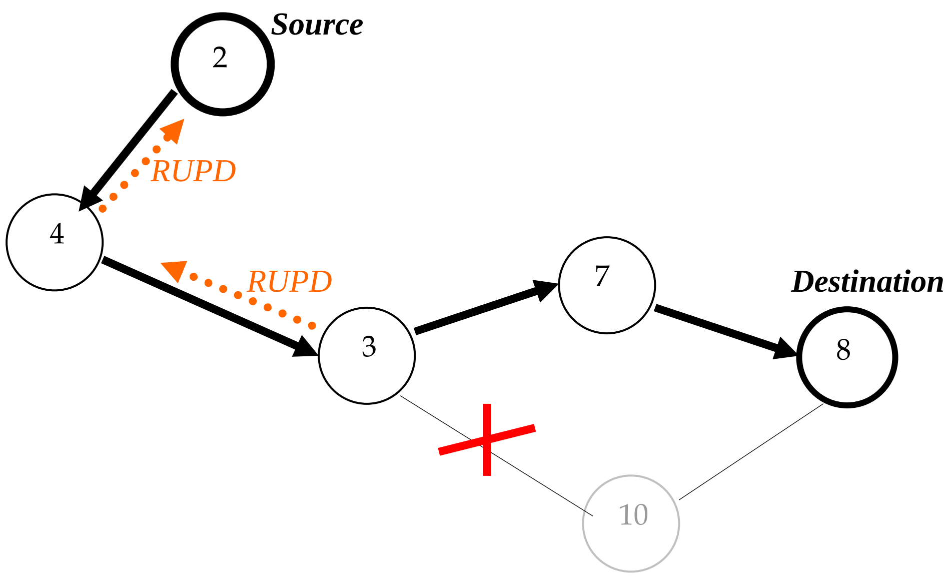


Fig (13) *RUPD message* from the node discovered broken link to the packet’s source