MANET with the Q-Routing Protocol

Ramzi A. Haraty and Badieh Traboulsi
Department of Computer Science and Mathematics
Lebanese American University
Beirut, Lebanon
Email: rharaty@lau.edu.lb, badieh.taboulsi@lau.edu.lb

***Abstract*--With ad hoc networks having much more advantages over other types of networks in a mobile world, this made it an attractive field for many protocols. In this paper, we propose an implementation of the Q-Routing protocol working over a mobile ad hoc network to enhance the performance of the packets sent and received. However, to implement a protocol in such an environment, many factors need to be taken into consideration, such as: exploration and learning over time to adapt to network changes. We used these factors in the protocol agents to update the routing tables found on each node accordingly. Our protocol has shown to perform well in MANETs as the load increased.**

Keywords-ad-hoc networks; exploration and learning; routing protocols.

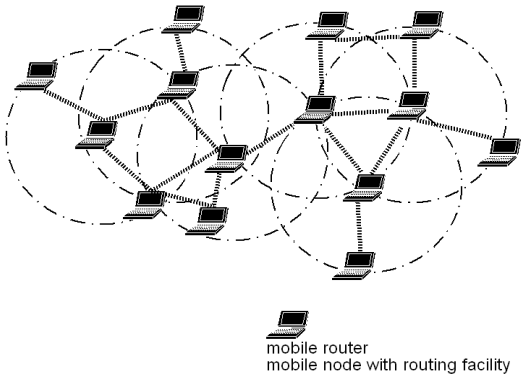
I. INTRODUCTION

As technology advances, people are gaining more interest towards mobile devices, which makes it an attractive source for many new enhancements. Mobile devices might range from mobile phones to portable design aids and laptops. The applications running on these devices most probably are not related to each other, and do not communicate. Nevertheless, there are scenarios where few devices move closer to each other forming a temporary network allowing transfer of different kinds of data and information. Such networks are known as *mobile ad hoc networks* (MANET) [1].

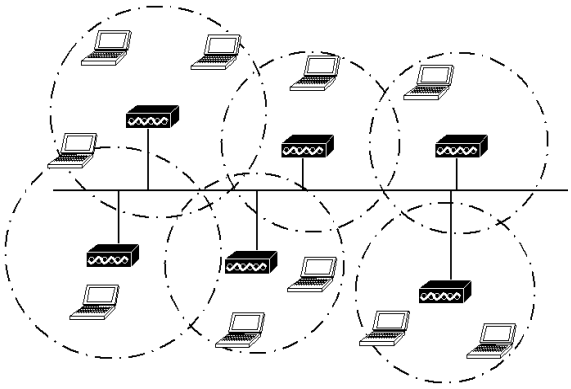
The main structure of the MANET, which distinguishes it from other networks, is that it can be formed without requiring any kind of infrastructure or administration. It contains mobile nodes that use an interface to communicate with each other wirelessly. These nodes can play the role of senders and receivers also known as *hosts*, or even *routers* that just forward the packets through. This gives a new ability to the network, nodes are no longer limited to the transmission range they got, and can connect to a mobile device, which is several hops away, and this is known as *multi-hop communication* [2].

Even though, as shown in Figure 1, ad hoc networks have different structure types, some networks share a hybrid of those two structures, such as the cellular network. Not to mention, the existence of a field of study to develop and simulate *wired-cum-wireless* protocols.

Ad hoc networks are not perfect, and they come with their own complications. Since the devices are mobile in nature, this means the network is ever changing. In other words, mobile nodes that were in range of each other, could leave the vicinity leading to disregard the direct link in between, and new nodes that were never near each other, come to be direct links. Direct link means it does not require multiple hops to reach the destination.



(a) Mobile ad hoc network



(b) Network with Infrastructure

Figure 1. Different network types of wireless structures.

In addition, different devices have different range capabilities, which mean a device might be able to reach another device, but this other device might not be able to reach the first one leading to an *asymmetric link*. A representation of an asymmetric link is shown in Figure 2. To solve this randomized nodes connectivity many different protocols were suggested [3] and will be subsequently discussed.

Q-Routing is the first routing algorithm to make use of reinforcement learning, called Q-learning [4][5]. Q-learning makes use of Q-values to perform updates and to estimate

 ISBN: 978-1-61208-183-0

how long it will take to send a packet to any particular destination through each one of the node's neighbors. In this paper, we propose an implementation of the Q-Routing protocol working over a mobile ad hoc network to enhance the performance of the packets sent and received.

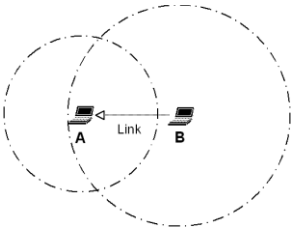


Figure 2. Asymmetric Link.

The rest of the paper is organized as follows: Section 2 discusses related work regarding mobile ad hoc networks and their associated routing protocols. Section 3 presents Q-Routing. Section 4 highlights the implementation. Section 5 presents the simulation results. Section 6 provides the conclusion.

II. RELATED WORK

Even though ad hoc networks do not require a specific structure, they still need a standardized way to base their communication at and make things work. For the nodes, in a mobile ad hoc network, to be able to decide on the route that the packets need to take to reach the specified destination, they need a convention, which is the *ad hoc routing protocol*. Moreover, at the start of the ad hoc network, the nodes are not aware of each other; thus, they need to discover each other by broadcasting to neighboring nodes their presence. Not only that, they also need to listen for other broadcasts in case a new node is added. The nodes might not only broadcast their own information, but are able to broadcast how to reach other nodes as well. Since there are many possibilities to design an ad hoc network, many types of related protocols specified for such network were discovered. Some of them are enhancements over others, and others are a combination of protocols.

A. Pro-active (Table-Driven) Routing

Just as its name suggests, this kind of routing maintains a table of the destinations and the paths to be taken, which is then broadcast to other nodes in the network. The time for convergence of the table varies between one and another depending on the algorithm used. Because of that time, it leads to a main disadvantage in case of a node dies or even a full restructuring of the network. The second disadvantage is that it needs loads of information for maintenance, which can scale up largely in big sized networks. However, while having a table of information, packets can be instantly sent according to the stored route reducing the latency for route discovery. The most common protocol used for this routing is *Highly Dynamic Destination-Sequenced Distance Vector Routing Protocol* (DSDV) [6].

B. Reactive (On-Demand) Routing

Unlike the pro-active routing, this type of routing only broadcasts for the destination when a send request is at hand. The broadcasted packets are basically Route Request packets, which are used to discover the path towards the destination. Since it does not store any information about the network, there will be a higher latency waiting for the path to be discovered. In case of a busy network, there might be too many broadcasts that can congest the network. On a highly mobile network, this routing works greatly, especially if nodes change frequently making any old route useless. The most known protocol used for this routing is *Ad hoc On Demand Vector* (AODV) [7].

C. Flow-Oriented Routing

Close to the reactive routing, this type of routing finds a route when demanded through a certain known flow. An example of this would be to consequently unicast whenever a new link is being advertised. With this comes its own consequences, which basically are taking too long while discovering totally new routes, and the probability of referencing to an existing traffic just to pay off for its lack of knowledge on the path. An example of this type of routing is the *Lightweight Mobile Routing protocol* (LMR) [8].

D. Hybrid (Both Pro-active and Reactive) Routing

This type of routing combines the best out of pro-active and reactive routing. It uses the pro-active routes stored when a routing is initialized and then uses the reactive broadcasting to deliver the packet to its destination. This gives the advantage of discovering better routes along the way. The disadvantage of this type is that its usefulness is related to how many active nodes are in the network. Also, it depends on how steep the traffic volume is. The *Temporally-Ordered Routing Algorithm* (TORA) [9] uses this type of routing.

E. Adaptive (Situation-aware) Routing

Just like the hybrid routing, this type uses both pro-active and reactive; however, the only difference is that it uses special metrics to decide which and when of the two models to be used. It shares the same disadvantages of the hybrid, and TORA is not just an example of hybrid routing, but also an adaptive routing.

F. Hierarchical Routing Protocols

This type of routing protocol designs hierarchical levels whereby pro-active or reactive routing is used depending on the level the node is in. It shares many of the Hybrid and adaptive properties. However, it differs in the decisions of which model to use depending on a specific attribute according to the level. This makes its advantages directly related to the depth of the levels drawn on the overall scheme. Another main disadvantage is that instead of depending on how steep the traffic volume, as is the case

 ISBN: 978-1-61208-183-0

with hybrid and adaptive routing protocols, it depends on the meshing parameters. One of the well-known protocols used is *Cluster Based Routing Protocol* (CBRP) [10].

III. Q-ROUTING

Q-Routing was proposed by Boyan and Littman [4][5]. They were able to come up with a new routing protocol that is based over reinforcement learning also known as adaptive routing over communicational networks. What most of other routing protocols focus on is finding the shortest route towards destination, but what many disregarded was taking into consideration the traffic load. Q-Routing was mainly developed as an enhancement to overcome the traffic load, which any network could fall into, through adapting to the current state the network is facing, and by that improving the performance. In Q-Routing, each node contains the algorithm for the decision making. They exchange information occasionally to update their stored data; one of which is the estimated time for delivery. Thus, the decision is made by taking the path that has the shortest delivery time instead of just the shortest path.

The Q-Routing policy works as follows: at first, every node stores an estimation of the time it will take the packet to reach all its neighbors. This is done by maintaining a routing table at every node making this protocol a pro-active one. The information that the table stores, is basically a combination of (y, d) where y is the neighbor and d is the destination. Their values imply the time taken for a packet to reach destination d passing by neighbor y. Assuming a node x wanted to send a packet to destination d. It will pass it to y since it has the lowest delivery time in its table, and then it will prompt y of the estimated time it takes the packet to reach d. After that, x updates the table information accordingly. Kardi Teknomo [11] wrote Q-learning pseudo code that can be used to enhance the Q-Routing algorithm when put in an ad hoc situation. If given a state diagram as an input with a defined goal, represented by a matrix R, the output is the minimal path from any state to its defined goal. This algorithm is mainly used by the agent that will learn through time, and by each state the agent either got a reward out of it, or none. This keeps going on until the agent reaches the given defined goal. The declared parameter γ has a range between 0 and 1. The closer it is to 1, the more the agent will delay the reward for a future better weight.

Since Q-Routing showed promising results with experimentations, other enhanced versions of Q-Routing where suggested such as the *Predictive Q-Routing* [12], *Dual Reinforcement Q-Routing* [13], and *Confidence-Based Q-Routing* [14].

IV. IMPLEMENTING A NEW MANET ROUTING PROTOCOL

There are numerous network simulators that can be used to help in simulating network scenarios and test performances - NS2/NS3, OPNET, NetSim, etc. In our work, we used Network Simulator 2, or NS2, for the features it gives, and since it is an open source which can be extended and modified. More than one version for the NS2 is out, but the one we used is ns-allinone-2.33 [15].

NS is an object-oriented, discrete network simulator, used primarily for research. It provides an important framework to simulate TCP, routing and multicasting protocols over wireless and wired networks [16].

Most of the studies made and developed around Q-Routing were involving wired networks; but, since the project is about Q-Routing over ad-hoc networks, things might need a few twists to make them work. Our implementation is based on work done by Francisco and Pedro [17].

As mentioned earlier, the main characteristic of mobile ad-hoc networks is that the nodes are in constant move, and so we should always adapt to the current new states of the network. To do so, we need a reinforcement learning procedure that displays the impact of a given action on the network. In our case, the action is the decision on which node the packet is to be sent next. The ideal situation is to have all of the packets sent from destination to source with a minimal cost. By cost we mean how much of the network resources were used.

Once a packet has been transmitted, it can either be delivered successfully or dropped. The packet is dropped when the *time to live* (TTL) has reached 0; it starts with a specific number and decreases at each hop from one node to the other. Once the packet reaches its destination, then it can be said that it was delivered successfully.

Q-Routing and reinforcement models require some value, say V, also known as a reward, which helps in the learning process. This value can be considered the sum of all the reinforcements starting at a specific state. Using the equations written by Bellman [18], the value of Q can be calculated based on the success or failure probability of transmitting a packet to the next hop. Therefore, every action done would affect the value of the V accordingly depending on how much the system did actually "learn" from the action done. So, if for example, the sent packet failed to be received by the target node, the value will decrease greatly; however, if it was received successfully, it will only drop a little. There are cases where the source becomes the destination at the same time. Doing so will not change the state of the value since nothing happened.

All the packets are built at the beginning state, and for each transition of the network state, estimation is stored at each node. Every node has two types of values, one is the optimal estimated V and the other is the V of its neighbors. This value will decrease as time passes from the time it started. The value of a node is not known for other nodes until a packet is sent. This is mainly to keep those nodes that do not interact with each other with a less value. Moreover, for each decision made by a routing table to transfer a packet through a specific node, disregarding others, will lead to a decreased value for those other nodes, which the packet did not pass through. The agents that are spread

 ISBN: 978-1-61208-183-0

through the network will act upon the information stored at the node they are currently residing. Once the agent arrives at the destined node, the value for that node will be broadcasted to the neighbors. Since it is always better to go backwards as well as forward with the updates, an update packet containing routing information will be sent from source to destination to keep both synchronized. With this bidirectional update, nodes that want to communicate with the source will have a better knowledge of it.

Exploring MANETs needs to have its own way, and the way we dealt with it is by using the Boltzmann distribution technique [19] to be able to identify as many of the actions as possible at first, and after that discovering whatever neighbors each of the nodes has to add them to the next-hops within the routing table. Since the network might scale up to become quite large, we do not want every agent to go exploring areas which it does not deem useful, and thus a greedy methodology could be used to identify these areas.

It is mainly impossible to set all of the media access control addresses as potential actions. Even if it was possible it would take a lot of unnecessary time. Thus, a new state needs to be added which explores the potential actions, and this is done by a broadcast. The action will be chosen using Boltzmann probabilistic technique. However, in case the agent does not have a specific action except to explore, then the exploration would be chosen as the action to be done, and it would be done by broadcasting to the neighboring nodes to discover them. Those nodes will most probably take it from there and continue to forward it to its destination. Since in mobile network exploration is the most important part of sending and receiving a packet, most of the network load should be dedicated for it. Therefore, there is no harm in using a greedy methodology to decide upon the next hops. The next hop is the node with a *V* that is greater than the one on the current node. Figure 3 demonstrates the idea.

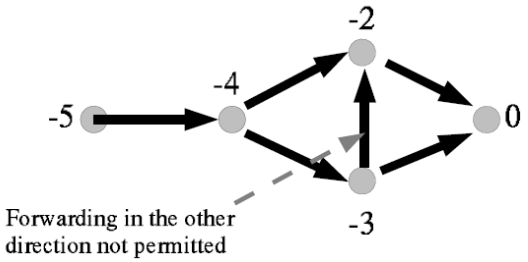


Figure 3. Greedy methodology.

The packets that are sent within the network go by the following format shown in Table I.

TABLE I. PACKET FORMAT

Field Name	Field Type	Field Contents
ORIGIN	IP Address	The original source of the packet
DESTINATION	IP Address	The final destination of the packet
SEQUENCENUMBER	Integer	Identifier generated from a counter at the source node
SOURCEVALUE	Floating-Point	$V_{src}(N)$
DESTINATIONVALUE	Floating-Point	$V_{dest}(N)$
HADERROR	Boolean	True if the packet's previous transmission failed
IPPACKET	Data Packet	IP Packet, or empty

The *routing agent table* has two variables, an id and a sequence number. The id is unique among all agents in the network. It is also able to produce distinctive indicator for the packets. The routing agent is connected to the routing entry and neighbor nodes in the sense that it keeps entries of all the interesting destinations, as well as, the neighboring nodes.

The *neighbor node table* has four variables, an id, a counter on how many sent attempts were made, how many were successful, and how many were received. Using this information, the estimation can be calculated.

The *routing entry table* has three variables, the id, the *V* for the destination, and the number of packets sent.

The *next hop route table* maintains two values whenever nodes advertise to each other a route. The values stored are the last known *V* and the time in addition to the table id.

The *forwarded packets table* stores the *V* of the node and the sequence number of the packets once it was forwarded. Since broadcasting packets to explore the network could lead to duplicates being generated, a unique sequence of numbers is maintained; therefore, if a node gets a packet twice, it will drop it.

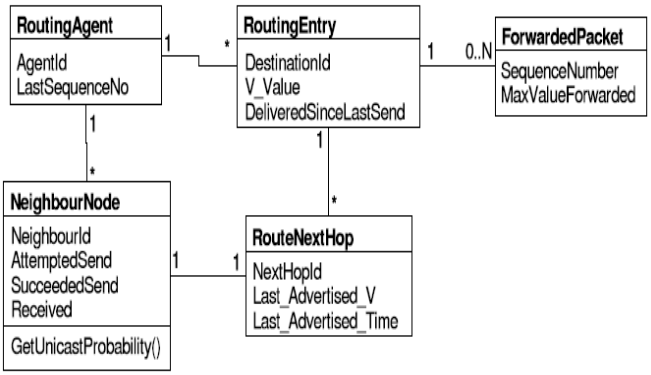


Figure 4. Table relations model.

The essential algorithm variables that we used are listed in Table II.

TABLE II. PROGRAM VARIABLES

Parameter Name	Units	Purpose
UNICASTSUCCESSREWARD	Reward	The Reinforcement received for successfully transmitting a packet. Fixed at -1.
UNICASTFAILUREREWARD	Reward	The Reinforcement received for a failed packet transmission. Fixed at -7.
EXPLORATIONUTILITY	Reward	The utility assigned to the exploration action
MINVALUE	Reward	The V-value of a broken route.
MINIMUMREWARD	Reward	Used to heuristically guide exploration
EVENTWINDOWSIZE	Time	The size of the window used to count events
EVENTWINDOWSAMPLES	Integer	Number of buckets to split event window into. Determines accuracy
PROBABILITYFROMRECEIVE	Float	Unicast Success Probability when have not attempted to transmit
RECEIVEWEIGHT	Float	How much Received Packets are weighted compared to Sent Packets
DECAYRATE	Float	How much the V-values grow every second that they are not advertised.
TEMPERATURE	Unitless	The temperature used in boltzmann action-selection
SEQUENCENUMBERMEMORY	Integer	Number of sequence numbers to record forwarded values for
MAXRECEIVESWITHOUTSEND	Integer	The number of packets that can be received on a flow without sending a response packet

V. SIMULATION RESULTS

We chose 50 nodes to be able to have enough fixed nodes while others are moving in different directions. Theoretically, the system should scale up as the number of nodes increases while still maintaining good results, but as the number of nodes rises up, the load on the network will increase in return; thus, affecting the total performance. For a detailed look at the simulation environment, users are referred to [20].

The trace file that was created out of the simulation was increasing greatly in terms of size due because the simulation included 50 nodes. The movement of the nodes was random. Since it was an ad hoc network, the 50 nodes were both clients and servers at the same time where they received, sent, and forwarded packets.

When the trace file was analyzed, we realized that there were a great percentage of packets received. The number of messages sent was relative to the time the simulation ran. Increasing the simulation time would increase the number of messages sent, but it should still maintain approximately 97% delivery rate, unless the network load increases gradually, then the delivery rate might start decreasing.

Messages Sent: 7385
Messages Received: 7233
Delivery Rate: 97.9417738659445

Next, the network load and the throughput of the Q-Routing protocol are calculated. The network load is basically the rate of the data sent by all the nodes. Whereas the throughput is the rate by which the nodes are receiving data. If we record the simulation to print out as it goes by, we end up with the graph, shown in Figure 5, for delivery ratio versus load.

Network Load = Packet size * Packets per Second * Number of Clients
Throughput = Network Load * Delivery Ratio

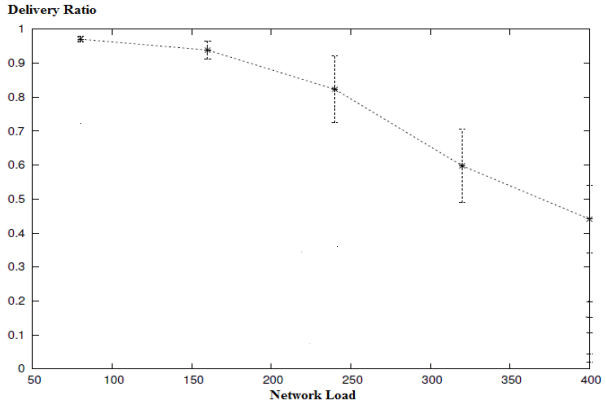


Figure 5. Delivery ratio versus network load.

As shown in Figure 5, while the network load increases, the delivery ratio decreases, and that is normal since more nodes will be congested leading to more packets being dropped. As for the throughput versus network load, we get the graph shown in Figure 6.

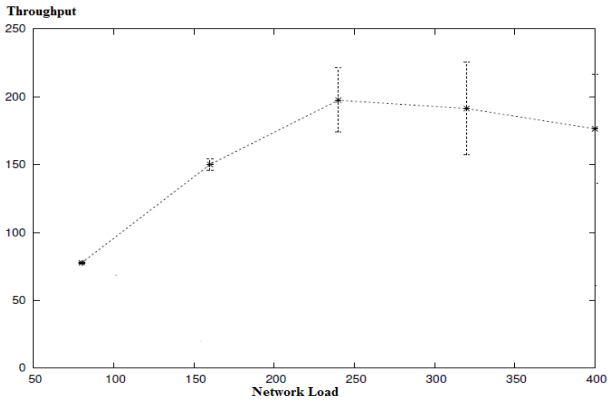


Figure 6. Throughput versus network load.

The throughput result shown in Figure 6 increases at the start, then slightly decreases as the network load starts to become large is due to the fact that at the beginning of the simulation, nodes start establishing connections with each other consecutively, and as the simulation time passes by, more nodes establish new connections while the previous ones still maintain their connection increasing the network load. After sometime, the network reaches a state where congestions occur at different nodes leading to a slight decrease in the overall throughput.

With the above results shown in Figures 5 and 6, we can say that the protocol was performing well not only when the

simulation started, but even later on when the load started increasing; thus, maintaining up to a 200 Kbps throughput. This is fairly good when compared to, for instance, what Jinyang Li *et al*. achieved in [21], where they stated that the maximum throughput achievable in an ad hoc network is 250 Kbps with packets having the size of 1500 bytes.

VI. CONCLUSION AND FUTURE WORK

When dealing with ad hoc networks, adding a little learning process to the way packets are sent and received will help out a lot, especially that the ad hoc networks are mobile making the exploration stage harder than usual. The mix between the Q-Routing and the wireless protocol led us to results which are promising. This protocol has proven to offer several advantages. For one, it is scalable as in no matter how many nodes we end up with, the number of agents will adjust accordingly. The second advantage is that there is not one main command, or central management system, where it leads the flows of the network; therefore, if any agent fails to do its work, it will not impact the overall reliability of the network. The third advantage is that agents are flexible in nature as in they can be modified according to any changes that the system may encounter.

As for future work, we plan to study other aspects of network performance such as latency and transmission rates. We also plan to implement the different types of Q-Routing and compare it with ours. In addition, the simulation we carried out was done only on 50 nodes, what we plan to do next is to analyze the protocol furthermore where more nodes are in place and more mobility around.

ACKNOWLEDGEMENT

This work was supported by the Lebanese American University.

REFERENCES

[1] S. Basagni, M. Conti, S. Goirdano, and I. Stojmenovic, Mobile Ad Hoc Networking, John Wiley and Sons, 2004, ISBN 0-471-37313-3.

[2] The MANET Web Page, Retrieved December 12, 2011, from http://www.ietf.org/html.charters/manet-charter.html.

[3] R. A. Haraty and W. Kdouh, "SDDSR: Sequence Driven Dynamic Source Routing for Ad Hoc Mobile Networks," Proc. of the World Automation Congress - International Symposium on Soft Computing for Industry. Budapest, Hungary, July 2006, pp. 1-8.

[4] J. A. Boyan and M. L. Littman, Packet Routing in Dynamically Changing Networks: A Reinforcement Learning Approach. Advances in Neural Information Processing Systems 6, San Francisco, CA, 1994. DOI: 10.1109/IPDPS.2005.323, pp. 671-678.

[5] J. A. Boyan and M. L. Littman, "A Distributed Reinforcement Learning Scheme for Network Routing," Proc. of the First International Workshop on Applications of Neural Networks to Telecommunications, 1993, pp. 45-51.

[6] C. E. Perkins and P. Bhagwat, Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers, *Comp. Commun. Rev.*, Oct. 1994, pp. 234–44.

[7] C. E. Perkins and E. M. Royer, "Ad-hoc On-Demand Distance Vector Routing," Proc. of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, Feb. 1999, pp. 90–100.

[8] L. Ji and M. S. Corson, "A Lightweight Adaptive Multicast Algorithm," Proc. of GLOBECOM '98, Nov. 1998, pp. 1036–42.

[9] V. D. Park and M. S. Corson, "A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks," Proc. of the IEEE Conference on Computer Communications (INFOCOM '97), Apr. 1997, pp. 1405-1413.

[10] C. C. Chiang, "Routing in Clustered Multihop, Mobile Wireless Networks with Fading Channel," Proc. of the IEEE SICON '97, Apr. 1997, pp. 197–211.

[11] K. Teknomo, Q-Learning Algorithm, Retrieved December 12, 2011, from http://people.revoledu.com/kardi/tutorial/ReinforcementLearning/Q-Learning-Algorithm.htm.

[12] S.P.M. Choi, and D. Yeung, "Predictive Q-Routing: A Memory-based Reinforcement Learning Approach to Adaptive Traffic Control," Proc. of the Neural Information Processing Systems, 1995, pp. 945-951.

[13] S. Kumar and R. Miikkulainen, "Dual Reinforcement Q-Routing: An On-Line Adaptive Routing Algorithm," Proc. of the Artificial Neural Networks in Engineering Conference, 1997, vol. 7, pp. 231-238.

[14] S. Kumar and R. Miikkulainen, "Confidence Based Dual Reinforcement Q-Routing: An Adaptive Online Network Routing Algorithm," Proc. of 16th International Joint Conference on Artificial Intelligence, 1999, pp. 758-763.

[15] M. Greis, Tutorial for the Network Simulator "ns", Retrieved on December 12, 2011, from http://www.isi.edu/nsnam/ns/tutorial/index.html.

[16] The Network Simulator – ns-2, Retrieved on December 12, 2011, from http://www.isi.edu/nsnam/ns/.

[17] F. J. Ros and P. M. Ruiz, Implementing a New Manet Unicast Routing Protocol in NS2, Technical Report, University of Murcia, 2004.

[18] R. E. Bellman, Dynamic Programming. Princeton University Press, 1957, ISBN 0-486-42809-5.

[19] D. Lindley, Boltzmann's Atom: the Great Debate that Launched a Revolution in Physics, The Free Press, 2001, ISBN-10: 0684851865.

[20] B. Traboulsi, Manet with Q-Routing Protocol. Master's Thesis. Lebanese American University. 2011.

[21] J. Li, C. Blake, D. De Couto, H. Lee, and R. Morris, "Capacity of Ad Hoc Wireless Networks," Proc. of the 7th ACM International Conference on Mobile Computing and Networking, 2001, pp. 61-69.

 ISBN: 978-1-61208-183-0