SIMULATED ANNEALING BASED HIERARCHICAL Q-ROUTING: A DYNAMIC ROUTING PROTOCOL

By

ANTONIO MIRA LOPEZ

Bachelor of Science in Computer Science

University of Science and Arts of Oklahoma

Chickasha, Oklahoma

2000

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 2007

SIMULATED ANNEALING BASED HIERARCHICAL

Q-ROUTING: A DYNAMIC ROUTING PROTOCOL

Thesis Approved:

Dr. Douglas K. Heisterkamp

Thesis Adviser

Dr. John P. Chandler

Dr. Nohpill Park

Dr. A. Gordon Emslie

Dean of the Graduate College

TABLE OF CONTENTS

LIST OF FIGURES

Figure Page

CHAPTER 1

INTRODUCTION

Internet is expanding at an incredible rate year after year. More efficient technologies need to evolve to keep up with the rapidly number of users that share the networks. With the increase in the number of users and the different application protocols such as ftp or http, the common problems in the communication among nodes will become more and more problematic. Some of these problems such as congestion create a huge impact in how fast packets can be delivered from one node to another node. Routers are the devices that manage how packets are delivered from a source to a destination. The internal algorithm that decides how the packet travels though routers is called a routing protocol. Routing protocols are critical when we think about how fast a packet makes it from source to destination. Their main purpose is to find an algorithm that provides the most optimal path from source to destination. Since networking is becoming larger and larger, this has become a really important research topic. There are some very well known internal routing protocols such as RIP, IGRP, OSPF that are commonly used in networks today. There is a lot of research being done in this area.

In 1992, two researchers named Littman M.L and Boyan J.A [5] worked in creating a routing algorithm that would combine the concept of routing with reinforcement learning.

A router must be able to realize when there has been a change in the topology or when a link in the network has failed. In these situations the router must be able to adapt quickly to the new topology by changing the way it is distributing packets to places that were in the path of that particular link. It must also do so when the network gets congested. The router must be able to efficiently and quickly change and routes packets in an efficient manner. There is a specific area in reinforcement learning that fits very well in the concept of routing. The concept is called Q-learning and this is the concept that Littman and Boyan used in their experiment to create Q-routing.

Throughout this paper the concept of Q-learning will be analyzed in detail. The different experiments and their results will be analyzed closely and they will be compared to the ones obtained in the experiments that will be conducted in the approach provided in this thesis. The objective of this thesis is to modify the already existing Q-routing algorithm so that it becomes more efficient. A new routing algorithm will be created and tested in the same environment as the one created for Q-routing. A recently developed algorithm that has improved the q-routing algorithm will also be described and compared against the proposed algorithm. The main problem that Q-routing has is that it does not perform exploration. In the proposed algorithm an exploration method will be explained and implemented. The results will be compared to those of Q-routing under different network loads.

The layout of this thesis will be done in the following manner: First an introduction of the important topics of this thesis such as Reinforcement Learning, Q-learning, Sarsa, Q-routing, and routing algorithms will be provided to the reader. Next, a more detail description of the experiments provided in Q-routing will be given. Then a

detailed description of the most advanced approaches done in the area will be given. Lastly, the provided approach will be described in detailed along with all the documentation needed to prove the difference in performance of the proposed approach vs. Q-routing [5] and the N Q-Routing Optimal Shortest Path [1] approach.

CHAPTER 2

REINFORCEMENT LEARNING

People earn most of their living experiences by trial and error. This means that in order to know whether an action is going to be worth it or not, they must try it first and see if it will yield a successful result. Reinforcement learning is very much based on this concept. It does not follow a specific set of actions in order to learn. It is not supervised. Other techniques such as Neural Networks use supervised actions to approximate more and more to an expected result. They are told what to do in order to achieve this based on previously known knowledge of the environment.

2.1 Components of Reinforcement Learning

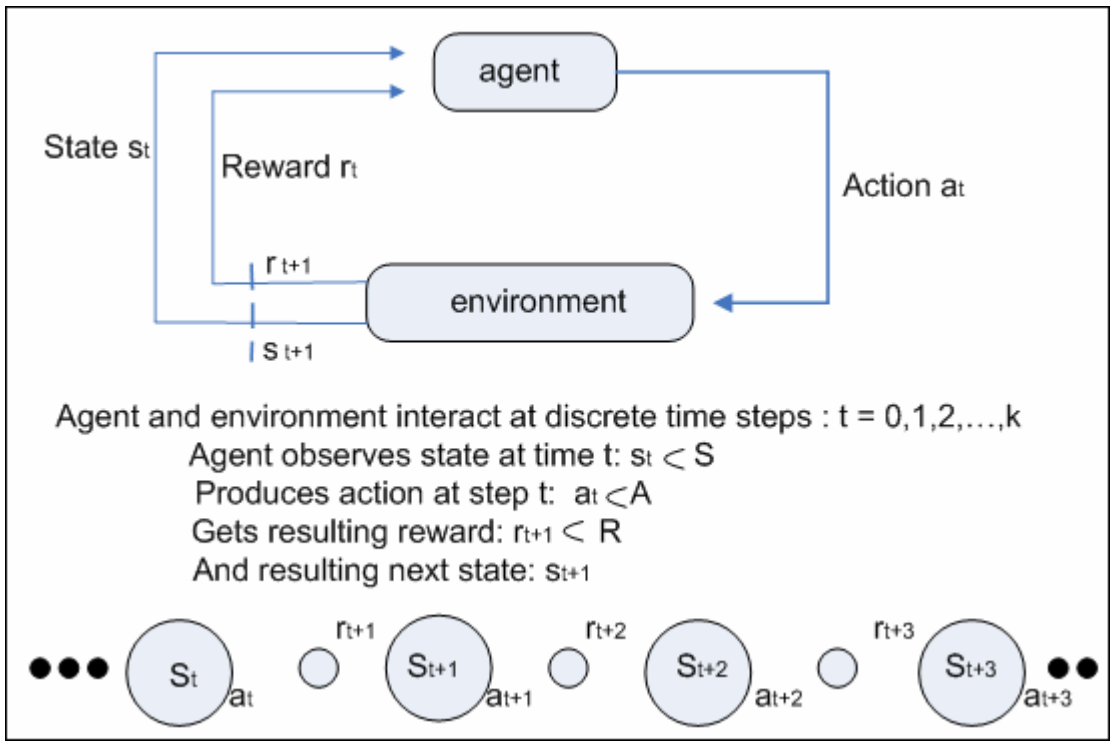

Figure 1 [11]: Diagram of Reinforcement Learning Setting

In Figure1 [11], we can see the different components that form Reinforcement learning. These components are the reward function, a policy, a value function, an agent and the environment itself.

The policy is the component that decides which action must be taken at a specific time in a specific state. "A *policy* defines the learning agent's way of behaving at a given time" [7]. The policy could be based on different things such as history of a list of actions to be taken given a specific situation.

The agent is the component that makes the decisions based on this policy. Once a state is reached the agent will make a decision based on the policy previously determined.

The reward function simply returns a positive or a negative value after performing an action. This is similar to the real life situation of trial and error and it is how we

humans learn to interact with new environments. If after taking an action we find it positive we will probably chose to execute it if we run into it again. If the result is negative we will most likely discard it unless we do not have any other option. In the concept of reinforcement leaning, the reward function returns a real number. It provides feedback to a state after a specific action has been taken. The goal is not to obtain a good positive value at each time step, but to maximize the sum of all the rewards at the final step [7].

The value of a state is the accumulated reward given to that state after following a specific policy and getting to a terminal state. An optimal policy would be one where starting from a random state we obtain the maximum possible reward when arriving to a terminal state.

The value function is the mapping of states to states values and it is found using a function approximator such as a neural network or a simple look-up table.

$$V^{\pi}(S) = E_{\pi}\{R_t \mid s_t = s\} = E_{\pi}\{\sum_{k=0} \gamma^k \, r_{t+k+1} | s_t = s\}$$

In the previous formula, E_{π} stands for the expected value when an agent follows a specific policy π. "We call the function V^{π} the state-value function for policy π" [7]. In the formula we can see that R_t is equal to $\sum_{k=0} \gamma^k \, r_{t+k+1}$, which represents the amount of weight given to future rewards. This concept is known as discounting. The agent will select actions in order to maximize the end result. In the formula the variable γ is a parameter in the range $0 <= \gamma <= 1$ called the discount rate. This parameter represents the present value of future rewards. As the value approaches 0, more weight is given to the recent rewards. When the value is closer to 1, more weight is given to the overall collected rewards [7].

2.2 Exploration vs. Exploitation

A very important issue in reinforcement learning is the concept of exploration and exploitation. Exploitation means that the agent will keep performing ways that it already knows that yield a valid solution but not necessarily the best one. Exploration is important since to a certain degree we must explore different paths in the environment to search for new paths that could yield better results that the once already known. Thus it is very important to achieve a balance between exploration and exploitation to achieve better results in experiments. “The agent has to *exploit* what it already knows in order to obtain reward, but it also has to *explore* in order to make better action selections in the future” [7]. It is very important for the agent not to get stuck in a local optimal solution. It must explore other solutions in order to find the global optimal solution. In the routing problem the environment is constantly changing very rapidly. In this type of environment we must adapt quickly to the new changing stages. Better paths may appear and without exploration the agent may not be able to find these quickly. Without exploration these paths may never be found and the system can deteriorate over time [20].

CHAPTER 3

TEMPORAL-DIFFERENCE LEARNING

Temporal Difference learning is based on two other techniques called dynamic programming and Monte Carlo methods. It combines features from both to create a more robust technique. Monte Carlo methods used experience to solve the problem. They do not require total knowledge of the environment in order to solve the problem. However we must wait for a number of transitions before we get a reward. The reward is based on episodes. The main idea is that we can make a prediction about a terminal state based on a current state and then update that prediction based on what happens in the states along the path. The simplest form of temporal difference is called TD(0). TD(0) can be written as $V(S_t) \leftarrow V(S_t) + \alpha\,[r_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$ [7]. In TD(0), the update is evaluated using $\alpha\,[r_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$ [7]. The learning factor is given by α. The value of γ in this case is 0 since we are analyzing the simplest case of TD(γ) .This means that we will not be taking into consideration the weight of the previous states and we are paying total attention to what happens in the current states. Figure 2 shows the TD(γ) algorithm.

1. Initialize V(s) arbitrarily, π to the policy to be evaluated
2. Repeat (for each episode):
    2.1 Initialize *s*
    2.2 Repeat (for each step of episode):
        2.2.1 a ← action given by π for *s*
        2.2.2 Take action a; observe reward, r, and next state, s'
        2.2.3 $V(s) \leftarrow V(s) + \alpha [r + \gamma V(s') - V(s)]$
        2.2.4 $s \leftarrow s'$
    2.3 until *s* is terminal

Figure 2 [7]: TD(γ) Algorithm

The figure shows how the updates are based on other updates. The variable γ will tell us how much weight will be given to the V(s') value or the state-value from the next state. The values for each state will be updated in this manner. If the value of γ is small the agent will be given more strength to the immediate reward and if the value is high the agent will give more strength to the overall collected reward. Temporal Difference uses the bootstrap that Dynamic Programming uses and the episodic update used in Monte Carlo methods. It is clear that by combining these features of these two methods, TD is more robust an efficient approach towards solving the reinforcement learning problem [7].

3.1 On-policy vs. Off-policy Methods

On-policy and Off-policy methods are used to make sure that an optimal greedy policy is used to make decisions. In order to achieve optimal policies we need to have a mixture of exploration and exploitation. As explained earlier, exploration simply means that we not always have to follow the greedy path in order to achieve optimality. Sometimes by exploring new paths we may actually encounter a more optimal solution. Exploitation on the other hand is based on always following the greedy solution which always guarantees to have an optimal solution, but not necessarily the most optimal one.

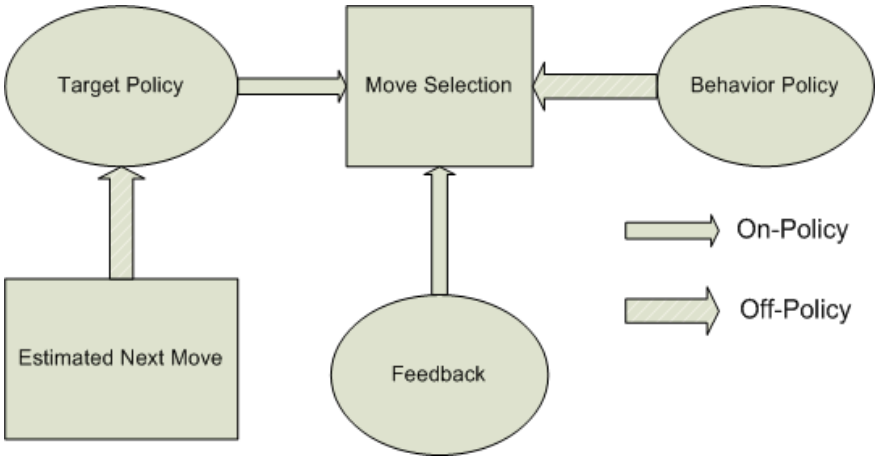

Figure 3 [23]: On and Off Policies

Figure 3 shows the difference between On and Off policies. When an update takes place in an on-line environment, the next move from one state to another is affected by the move itself. When we deal with an off-policy, the evaluation of the move made in a state will affect the policy from the current move but the move itself is independent of the policy update and therefore we can use a different policy to make the move. On-line policies depend a lot on exploration in order for the action values to be accurate [23]. Whereas On-line policies are based on a combination of exploration and control, Off-line policies separate the two of them. In the next section, two different algorithms will be presented. The first one is called Q-learning and it based in an off-policy environment. The second one is called Sarsa and it is based on an on-line policy environment.

3.2 Q-Learning

Q-Learning is an example of an Off-Policy Method. The method follows the follows the algorithm presented in figure 4 [16] to perform the update. In the algorithm the variables *s* and *s'* represent the states and the variables *a* and *a'* represent the actions. Q represents the state-action value and α and γ represent the learning rate and the discount factor:

1. Initialize $Q(a, s)$ arbitrarily
2. Repeat (for each episode)
2.1 Initialize s
2.2 Repeat (for each step episode) until s is terminal
2.2.1 Choose a from s using policy derived from Q (e.g. ε-greedy)
2.2.2 Take action a observe reward r, state s'
2.2.3 $Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$
2.2.4 $s \leftarrow s'$

Figure 4 [16]: Q-Learning Algorithm

In Q-learning, the model of the system is learned in terms of Q-values. Each value is of the form *Q(s, a)* which represents the expected reward of taking an action *a* in state *s*. In state *s*, if Q-values are learned to model the system accurately, the best action is the one with the highest Q-value in the vector *Q (s, x)*, where *x* represents the set of actions, *a*, that can be chosen from state *s*. Thus Q-learning first learns a representation of the state of the environment in terms of Q-values and these values are the ones that will be used to make the control decisions.

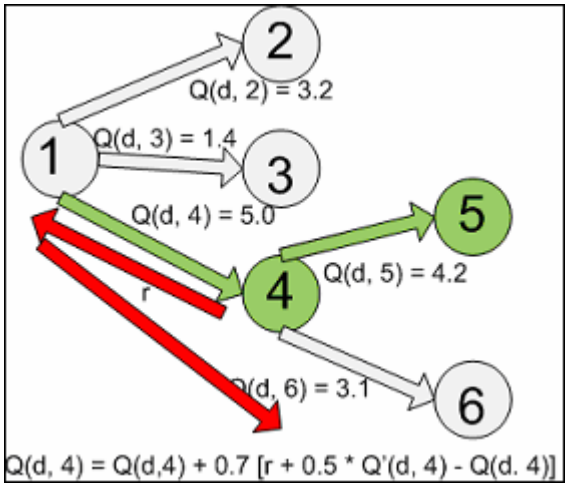

Figure 5: Q-learning example

The example in figure 5 illustrates a basic step in the Q-learning algorithm. In the example the learning variable α has the value 0.7. The discount factor variable, γ, has the

value 0.5. The variable d represents the destination node. Node 1 must choose a neighbor node such that its value is the maximum according to the policy. Node 4 has the highest Q value and therefore is chosen. Node 4 will then send the reward value back to node 1. Once node 1 obtains the reply from node 4 it updates its new Q estimate Q(d, 4). The new state would now be node 4. Node 4 must process the same computations that node 1 executed. This process will repeat until the destination node is reached. At this point a new episode will take place. As more episodes are executed the Q values will approximate more and more to the desired Q values.

CHAPTER 4

ROUTING ALGORITHMS

Routing is the process of deciding which path is more suitable and efficient to send a packet from a source to a destination and traveling through an unknown number of routers to achieve it. Packets carry a field in their header called destination field which carries the IP address of the destination device. A router must look at this field and decide which of the next available hop will be picked to efficiently deliver the packet to its final destination. In order to do this, a router must look at its router table. The main problem with routers takes place in dynamically changing networks where a link failure can affect the performance of the entire network. Static shortest path first would be considered a static routing, where an administrator has to configure the network routing and change it manually in case a failure or any other negative event takes place in the network. On the other hand, Q-Routing is considered dynamic routing, specifically a type of distance vector routing, since it looks at its local neighbors in other to make a routing decision. Dynamic Shortest path is also a routing algorithm that uses Dijsktra's algorithm to figure out the most optimal path to send a packet from source to destination.

There are some important concepts about routing that are important to mention since they are important parameters used in the experiments. The first one is Packet Delivery Time

(T_D) [15]. Packet delivery time stands for the total time that a packet spends since it is first introduce in the network at the source node until it is consumed at the destination node. This will be the main metric that will be used in the experimentation. The second one is the waiting time at intermediate queues (T_W). This refers to the time that a packet may be waiting at the intermediate node's queues before being consumed at the destination node. Transmission delay over links (T_X) is the time it takes for a packet to travel through the transmission link from one node to another. Thus we have that:

$$T_D = T_W + T_X$$

This formula will be used later since it is important to take these parameters into account when updating the Q-values in the environment [15].

4.1 Shortest Path Routing

Shortest Path is one of the simplest routing algorithms. It simply delivers packets from source to destination using a metric to figure out the most optimal path. The way it is done is by using a metric (weight) between the links connecting the nodes and using the one that yields the minimum collected weight sum of the links.

The Shortest Path Routing algorithm works by performing Dijsktra's algorithm using routers which represent the edges in a graph. The shortest path is then executed using a metric such as the smallest number of hops from the source to the destination or the bandwidth of the links connecting the routers. The algorithm can be performed using any other metric. The algorithm tries to find the most optimal path using that metric from the source node to the destination node.

This creates an important problem when the network experiences congestion. In figure 6 we can see that the shortest path to Router 8 from Routers 1, 2 and 3 is going

through link Router 4. If the link gets congested these routers will not make any changes since they will always compute the shortest path based on the hop count from source to destination. The queue in router 4 will quickly get full and packets will begin to be dropped. These packets will be retransmitted using the same path and the network load will increase very rapidly. Router 4 will create a bottleneck in the router and the only way to solve the problem will be by manually changing the routing paths in the network.

Although the Shortest path algorithm is very easy to implement, it becomes very inefficient as the network load increases and the queue of the links get full [17].

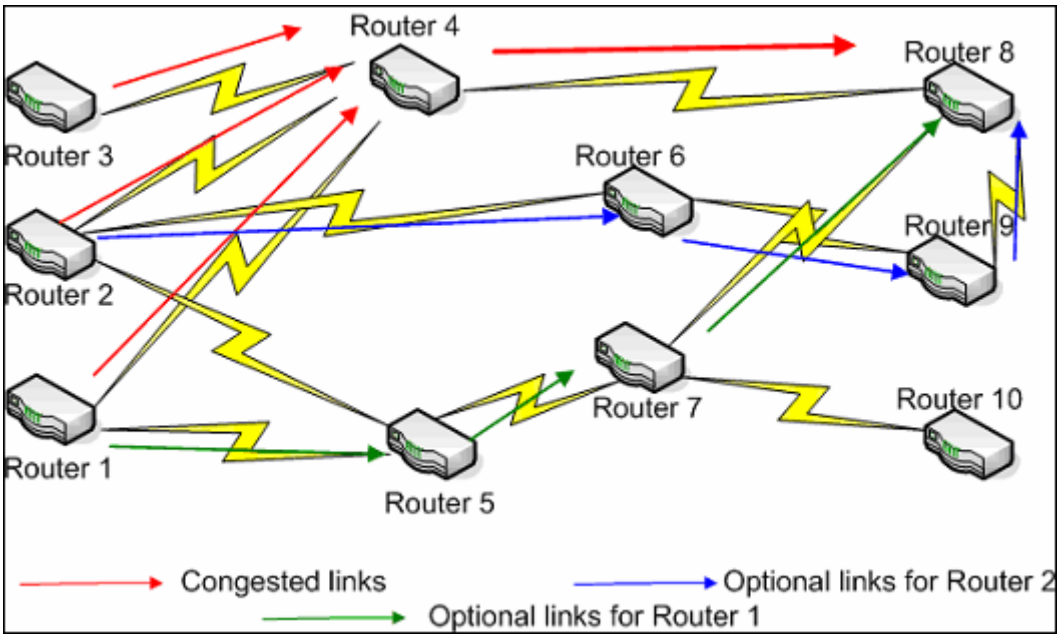

Figure 6: Shortest Path Problem

The optimal solution to this problem would be some type of learning algorithm that would tell the Routers what link to use under different network environments. This is what the Q-Routing algorithm accomplishes. It makes use of reinforcement learning so that each node learns partially about the environment and is able to make smart decisions under different network environments.

4.2 Q-Routing

Q-routing [5] was developed by Littman and it is the first one to make use of reinforcement learning. Q-routing is based on the concept of Q-learning. Q-learning makes use of Q-values to perform updates. Similarly, Q-routing uses Q-values which estimate how long it will take to send a packet to any particular destination through each one of the node's neighbors. The state s in the optimization problem of network routing is represented by these Q-values in the entire network [5].

In Q-routing, each node contains a packet-routing module. The module performs actions such as routing messages arriving at a node to a neighbor with the smallest total delivery time. When a packet arrives, it sends an estimate of the remaining delivery time back to the node that sent it. When receiving a reply from a neighbor, it updates the estimate for that neighbor.

The Q-routing algorithm can be stated as followed:

```
While (true)
        Select a packet from the queue and let d be the packet destination
        Select y', such that y' has minimal Q(y,d) from all neighbors y of the current nodes
        Wait for a reply from node y'
        Update Q(y', d) using the new estimate from y' such that
        Q(y', d) = Q(y', d) + α[Qy'(z', d)+ t + q - Q(y', d)]
End while
```

Figure 7[6]: Q-Routing Algorithm

A packet arrives at a router and it is then processed. In the algorithm, this router would find itself in state s. The agent must select a neighbor y' such that y' has minimal Q(z', d) from all neighbors y' of x and return the reward back to s. The Q-value for the chosen neighbor is then updated. The update rule used in the Q-routing algorithm is very similar to that used in the Q-learning algorithm. In the update rule, *t* represents the

transmission delay, the time it takes for a packet to travel over the media to the next node. The variable q represents the time the packet spends waiting in the queue. As mentioned earlier, these variables are needed since in reality these two factors increase the time of the packet making it to destination. In the update equation, $Q_{y'}(z', d) + t + q$ represent the new estimate and $Q(y', d)$ represents the old estimate. As mentioned in the previous algorithms, the variable α represents the learning factor. In many reinforcement learning problems this factor gets reduced as the optimal policy is reached and the environment is finally learned and stabled. Research in this specific problem has proved that this is not the case. The variable α must always maintain a high value since the network is constantly changing and therefore constant learning must be taking place. In the work conducted by Littman and Boyan, a value between 0.5 and 0.7 was used in all of their research experiments [5].

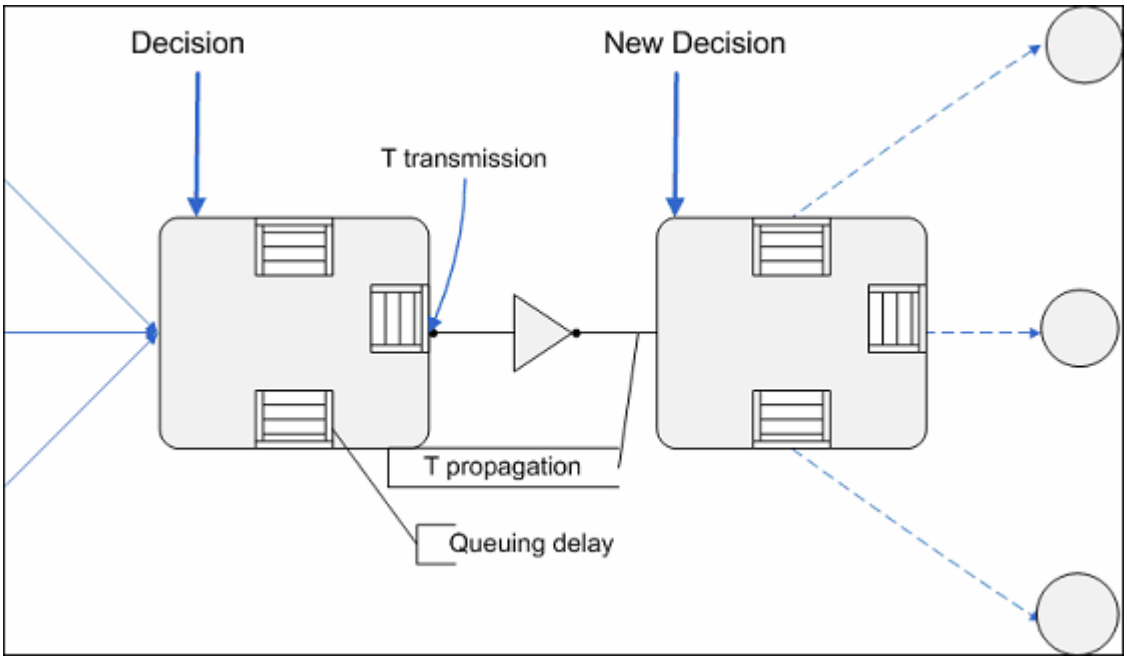

Figure 8 [10]: Computation of the immediate cost

In Figure 8 we can see the steps that a packet goes through to travel from node to node. First it waits in a waiting queue. When it is ready, it is transmitted and propagated through the media link.

The previously mentioned algorithm is purely greedy. It completely exploits the environment to search for the most optimal paths. For this type of scenario, where the environment is constantly changing we need to perform some sort of exploration to avoid getting stuck in a local maxima. In following sections, new approaches to the Q-routing algorithm will be described. These approaches will make use of exploration and other techniques in order to overcome the main problems found in Q-routing.

4.2.1 Q-Routing Results

The Q-Routing experiments were conducted under an irregular 6x6 grid topology. A discrete event simulator was used to model the motion of packets through a LAN [7]. The experimentation was executed after learning had been done for the Q-routing method. Then, packets were tested from sources to other destinations under different loads using both, Shortest Path and Q-routing and the results were compared. A router simulator developed by J. Boyan and M. L. Littman was developed in a C compiler to conduct the experiments. The compiler works under the WISH shell using the Solaris operating System.

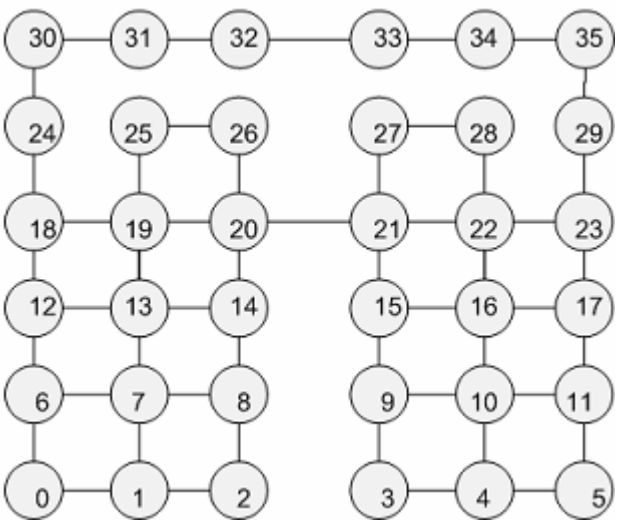

Figure 9 [24]: 6x6 irregular grid

As we can see the irregular grid is divided into two areas. The left area is composed of nodes 1 to 18. The right area is composed of nodes 19 – 36. The two areas are only able to communicate with each other using either the link that goes from node 18 to 19 or the link that goes from 12 to 25. We will refer to link 12 to 25 as R1 and link 18 – 19 as link R2. Communication between the two areas takes place mainly through link R1. In the experiment when this link becomes a bottleneck because of increasing loads, it is important for the adaptive algorithm to be able to switch to link R2 [24].

In their experiments, Load level represents the probability of a node generating a packet at any simulation time step. “Time is interpreted as simulation time which is represented by discrete time steps synchronized for all nodes in the network” [15]. These packets are introduced in the network periodically in a random node. The average number of packets in the network at any simulation time step is n x Load level, where n is the number of nodes in the network [15].

Littman and Boyan compared their Q-routing results with the Shortest path algorithm. In their experiment results they showed that Q-routing performed much better

under high load conditions than the Shortest Path algorithm. As mentioned earlier the Q-routing algorithm used total delivery time from source to destination as its metric.

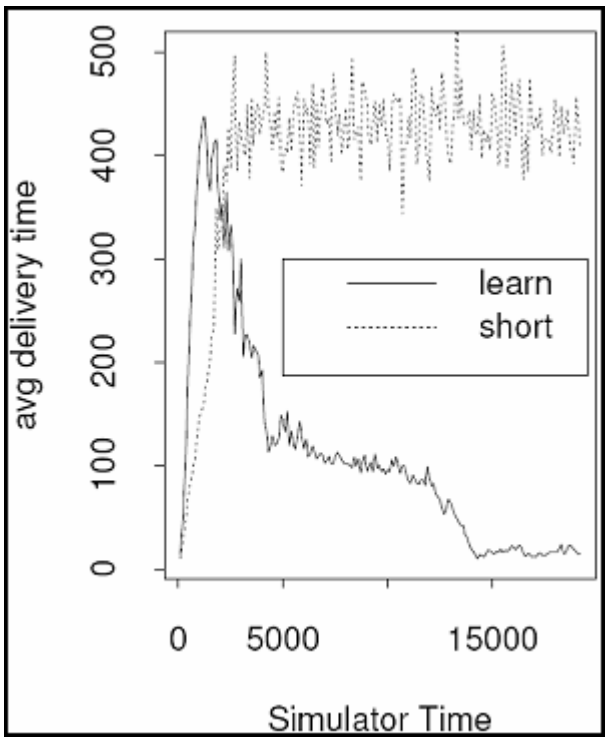

Figure 10 [6]: High Network Load

In Figure 10 the x axis represent the simulator time and the y axis represent the average delivery time of packets. The figure shows that at the beginning of the simulation Q-routing takes some time to learn the environment and therefore performs worse than Shortest paths. As soon as Q-learning updates its Q-values and has more knowledge of the environment, the average delivery time becomes rapidly lower than that of Shortest paths. Q-routing finally converges to an optimal environment. It does not take much time for Q-routing to perform better than shortest path under these network conditions.

When the network has a low load the situation differs. Shortest path performs optimally under this situation. Since the path that is taken is optimal and there is no

congestion in the network, the packets never wait in the queues to be delivered and delivering packets to their destination takes place very rapidly.

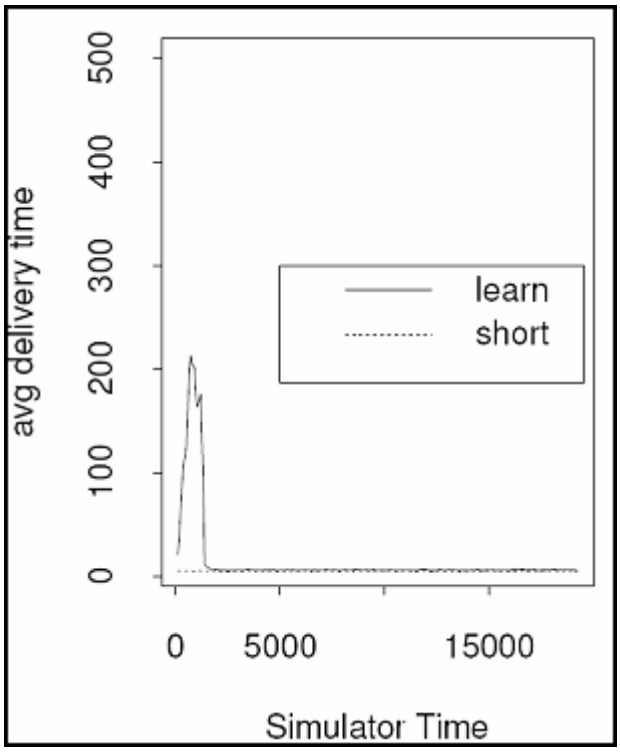

Figure 11 [6]: Low Network Load

Figure 11 shows that the shortest path algorithm performs optimally the entire simulation time. Q-routing on the other hand needs some time to learn and it does not perform efficiently during this time. Eventually it learns the environment and converges to an optimal stage.

Thus we can see that the main problem that Q-routing has is that under low network loads, the learning phase makes it very inefficient since packets travel for a while through the network in a very inefficient manner. The other issue that has been under a lot of research is making the learning process more efficient. There have been new studies that have introduced new ideas and factors into the Q-Routing algorithm that have made it a lot more efficient and robust.

4.2.2 Recent Advances in Q-routing

Since Q-Routing was first introduced by Littman and Boyan in 1994, a lot of research has been done in the area. There have been a lot of new approaches that have made changes or introduced new ideas into the Q-Routing to overcome its main problems and to make it more efficient and robust. Some of the most significant advances to Q-Routing were done by algorithms such as Confidence based Dual Reinforcement Q-Routing [8] and N Q Routing Optimal Shortest Path (NQOSP) [1]. This thesis will focus mainly in the NQOSP algorithm since it is more efficient and has recently been developed. Confidence based Dual Reinforcement Q-Routing was developed by Shailesh Kumar and Risto Miikkulainen in 1997.

4.3 NQOSP – N Q-Routing Optimal Shortest Path

The N Q-Routing Optimal Shortest Path algorithm [1] was created by Abdelhamid Mellouk, Said Hoceini, and Samia Larynouna in 2006. This approach requires that the routers maintain a link state database that contains information about the topology of the network. A routing protocol such as OSPF continuously sends these types of packets between routers to exchange information about the network. In this case, the algorithm only exchanges these messages when there is a change in the network such as congestion. Once the routers have the updated information about the topology of the network, the N-Optimal optimal paths algorithm is computed and the router determines the most optimal path.

The way the N-Optimal Paths is constructed is by using Dijkstra's algorithm. It is basically a modified version of the Shortest Path algorithm. A modification of the algorithm is made to avoid loop paths in the network.

The algorithm also makes use of the Q-Routing approach. They make some modifications such as not taking into account the transmission time from one router to another since they consider that this factor does not really have a big effect in the routing process. The reinforcement signal T, following the Q-routing algorithm, is described as:

$T = min\ y \in_{neighbor\ of\ x} \{qy + Q(y, x, d)\}$ [1],

where $Q(y, x, d)$ indicate the estimated delivery time for a packet to go from router x through router y to the router destination d. Once that packet reaches router y, the router will determine the optimal path to send the packet. Once this decision is made, router y puts the packet in its queue and send the reward value, T, back to router x. Router x then updates its information in the following way:

$\Delta Q(x, y, d) = \eta\ (\ \alpha + T - Q(x, y, d))$ [1]

Thus the new estimation Q'(x, y, d) would look like this:

$Q'(x, y, d) = Q(x, y, d)\ (1 - \eta) + \eta(T + \alpha)$ [1]

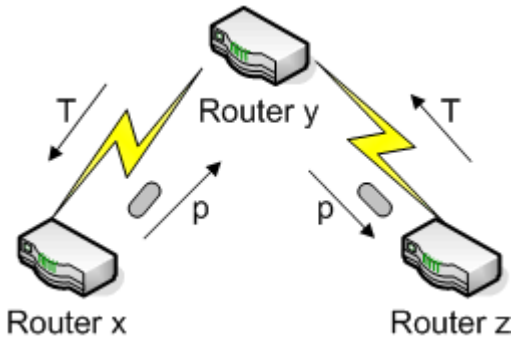

Figure 12 [1]: Updating of the Reinforcement Signal

The algorithm N Q-routing Optimal Shortest path was compared to the standard Q-Routing under different load conditions. In their experiments, the authors compared NQOPS to other routing protocols such as Shortest Path, RIP and Q-routing. In my experiments I am interested in comparing Q-routing performance as well as the

performance of the NQOSP, thus the results obtained in the RIP routing algorithm will not be analyzed.

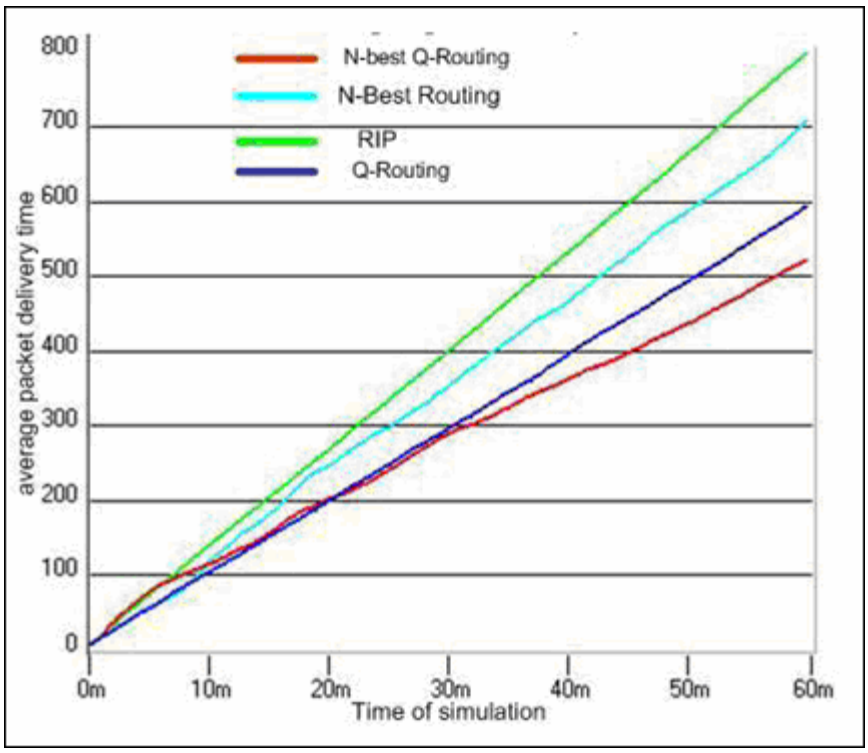

Figure 13 [1]: Network with continuous high load

As we can see in Figure 13, under high network loads, NQOSP, converges faster to an optimal environment than Q-routing does. The mean of the delivery time in NQOSP is reduced by 25% in comparison to that of standard Q-routing [1]. The picture represents a networking scenario where the network load is constantly increasing and therefore the average delivery time converges to infinity. In reality, the network usually suffers peaks of congestion. A more realistic scenario would be more like the one shown in figure 15.

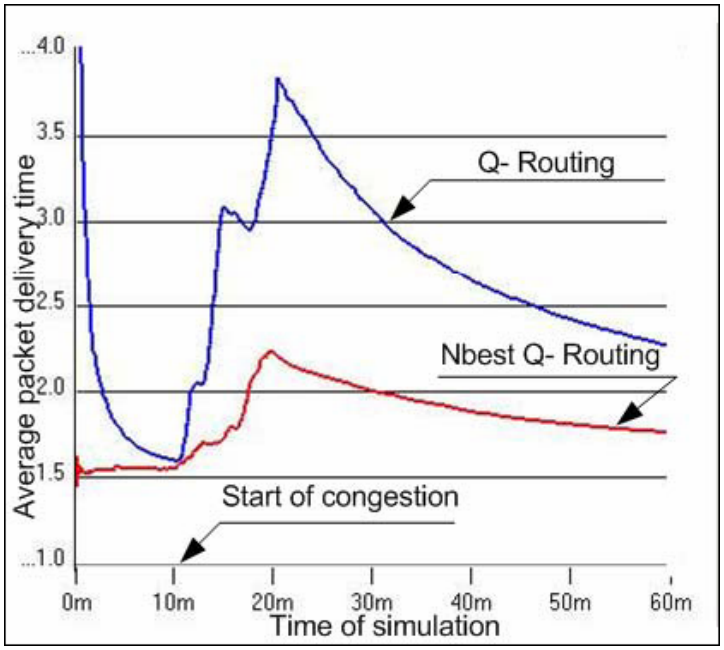

Figure 14 [1]: Network with a peak of high load

In figure 14 we see that in an environment where the network is at a low level and all of a sudden we have a peak of high load. After the peak of high load, the network load starts decreasing. NQOSP adapts much quicker to the change in the environment than standard Q-routing. Besides the fact that NQOSP learns quicker than Q-routing, NQOSP has the capability of avoiding loops and Q-routing does not.

CHAPTER 5

Q-LEARNING BASED ON SIMULATED ANNEALING

As mentioned earlier, the balance between exploration and exploitation plays a crucial role in all reinforcement learning problems. In 2004, Maozu Guo, Yang Liu, and Jacek Malec, created a new approached called the Q-learning Algorithm based on the Metropolis Criterion. The main objective of this approach is to improve the already exiting ε-greedy method by reducing e during the learning process. This is done though the use of the simulated annealing (SA) algorithm, in which a "local change of a solution to a combinatorial optimization problem is based on the Metropolis Criterion" [3]. In this algorithm there exists a transition probability P(i (j) of the metropolis criterion which indicates whether or not the transition from state i to j will take place. P(i (j)) can be defined for a maximization problem as follows:

$P(i \rightarrow j) = 1$, if $f(j) >= f(i)$ [3]

$P(i \rightarrow j) = \exp((f(j) - f(i)) / t)$, otherwise [3]

Where f(i) and f(j) are the values of the cost functions of the optimization problem. The pseudo code for the proposed SA-Q-learning approach is as follows:

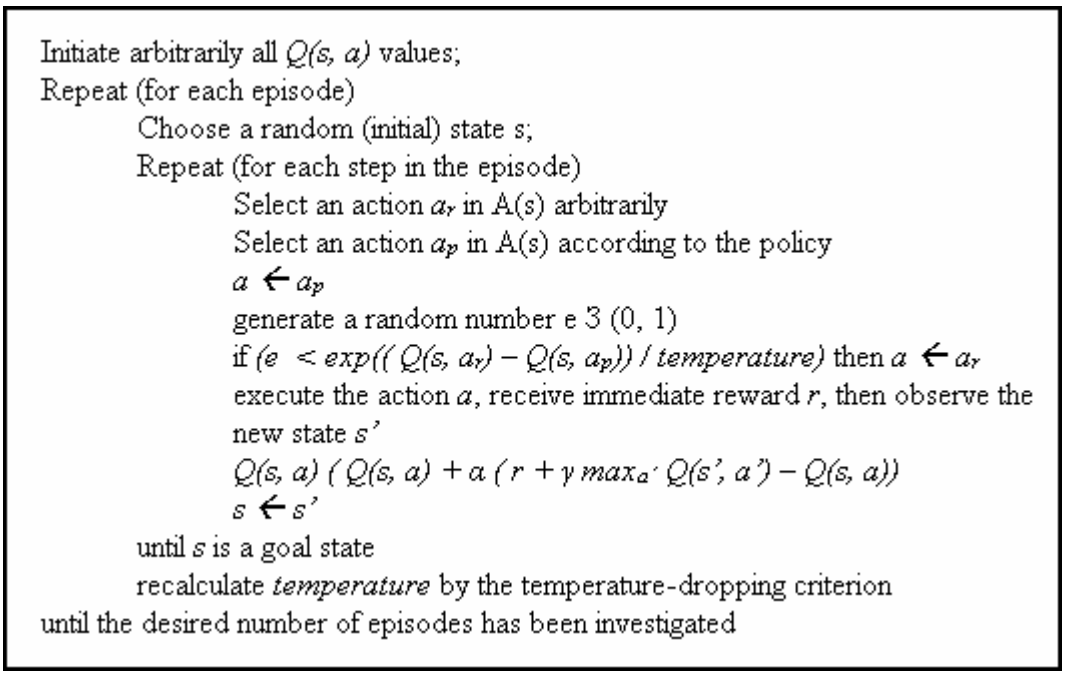

Figure 15 [3] : Simulated Annealing Algorithm

The temperature-dropping criterion can be computed arbitrarily. One of the most common ways is to compute it using the geometric scaling factor criterion, i.e., $t_{k+1} = \lambda\, t_k$, where k = 0,1,2,… and $\lambda \in (0.5, 1)$. The closer the λ value is to 1, the slower the decay will be in the algorithm [3].

CHAPTER 6

PROPOSED APPROACH: SIMULATED ANNEALING BASED HIERARCHICAL Q-ROUTING

6.1 Overview of the Proposed Method

The proposed algorithm is a hybrid approach from several research approaches that I have read and with the purpose of overcoming the problems that Q-routing presented. The main problem that Q-routing had when it was not initialized to shortest path was that it took some time to learn the environment so the algorithm performed very poorly at initialization. In the proposed algorithm this information will only be obtained at the initialization of a router. The router will obtain a map of the network at the time that it is initialized. This way the Q values will be initialized to choose the neighbor that provides the shortest path to the destination according to the bandwidth metric and with a loop free environment. At initialization an algorithm will be performed in the network to segment the space. This is mainly done so that some agents can efficiently manage local spaces of the network and others will be able to have a more global view of the network to make better decisions.

After initialization a router will follow the Q-routing algorithm in order to transfer packets across the network. When a packet is inserted in the network, the global agent in the router will look at the destination packet. It the packet is in the local area, the packet will be passed to the local agent who basically will perform basic Q-routing policy to

transfer the packet to the destination. If the destination packet happens to be in an external area, then the global agent will choose a border router in the area as the local destination. This is done by encapsulation. Once the packet arrives to the border router, the local destination will be discarded and the border router will then have to see if the destination is in an adjacent area or in a non adjacent area. If the destination is going to an adjacent area, the border router will have information about adjacent areas and therefore will be able to efficiently send the packet to its final destination. If the destination is in a non adjacent area, the border router will once again choose the border router belonging to an adjacent area that yields the minimum transmission time to the destination area. This process will be executed repeatedly until the packet arrives at its final destination.

The main idea is that we subdivide the task of transporting packets in areas. This way the information within areas will be updated very often and the Q-values will be kept updated. The global agent will also be able to have a better global view of the network and therefore it will be able to explore in a more efficient way whenever is needed. Exploration will be done by using Simulated Annealing. This term has been explained in chapter 4. For Simulated Annealing to work efficiently, a global value must be used that will help us keep the network stabilized. This is important especially at environment changing times. Each router knows how many packets are at any specific time in its queue and this value basically tells the router the rate at which it is being congested. A threshold can be then used to trigger a router to send a congestion message to border routers in adjacent areas so that some of these routers start sending packets through other border routers in order to avoid more congestion. The trigger takes place according to

congestion levels. Every time a packet arrives at a router its waiting queue utilization is computed. The congestion level changes every time a determined percentage in utilization change is reached. When the change takes place, this value is then sent to all other routers in the area and they will update their Router Congestion Table accordingly. This is the value that will be used to perform exploration. This percentage is the one being used as the temperature. All routers maintain a table about the congestion values of all other routers in the area. This value is initialized to 0 and will only be updated when another router reaches a specific change in its queue utilization. In order to be able to send these packets quickly these routers will use quality of service to provide higher priority to these packets used for congestion management purposes.

In the following sections the proposed method will be explained in a lot more detail. How segmentation is applied to the problem will be explained first. Secondly a detailed description will be given about how decisions are based on the hierarchical model of the network. Following that, there will be an explanation about how simulated annealing is used to avoid the congestion problem. The application of quality of service in the routers to speed up transfer of congestion messages in an area will also be presented in this section. Lastly the results of the proposed method will be presented along with a performance comparison with the Q-Routing algorithm under different network loads.

6.2 Network Space Segmentation

Network Space Segmentation is the mapping of routers to areas in the topology. The main idea is that in large graphs is very difficult to maintain global information without having to exchange local information to every node in the network. To avoid this

we could select areas and within these areas we could select border nodes to collect information about other adjacent areas and partial information about further areas in the network. This way a node will have enough information to make a better decision to transfer a packet to its final destination. It will also be able to perform exploration at times when the network may be very congested in order to avoid more congestion.

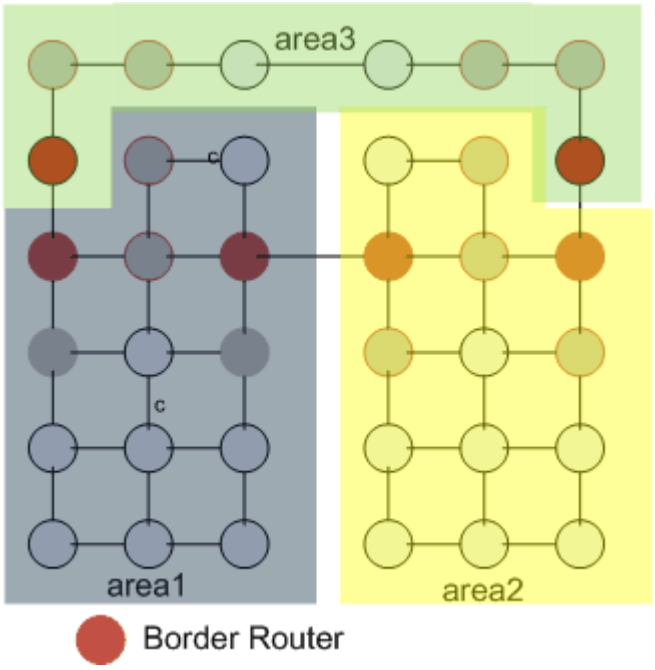

Figure 16: Network Segmentation

The key to efficient segmentation has to do with finding links that could potentially create bottlenecks in the network and making these the boundaries of the areas. The ideal scenario would be that an efficient algorithm would be able to dynamically find these boundaries and assign and modify areas dynamically as the network topology changes. This section is included as future work later in this thesis. The way this step has been implemented in the proposed method is manually. The administrator of the network is the one responsible for segmenting the network at this point.

Segmenting the Network Space has multiple advantages. The main problem with Q-Routing is scalability of large networks. As the network size increases, the needed Q table storage for each router becomes impossible to maintain. By segmenting the Network Space we assign roles to different routers. The local routers only need to have Q-values about those routers in their local area. The border routers are the ones that will have more Q-values. These routers will have the Q-values for routers in their area and routers in adjacent areas. Border routers will also maintain Q-values for external areas but they do not need to store information about every router in those areas, just the area itself.

Segmenting the network space also adds security since now we can create areas that belong to different entities with different roles and create security to access them through the border routers. Network security has not been contemplated in this paper but it could definitely be added in future work.

6.3 Hierarchical Network Model

Once the Network space has been segmented, areas can be used to make local and global decisions. Within an area a local agent will perform Q-Routing to send a packet from any source to destination.

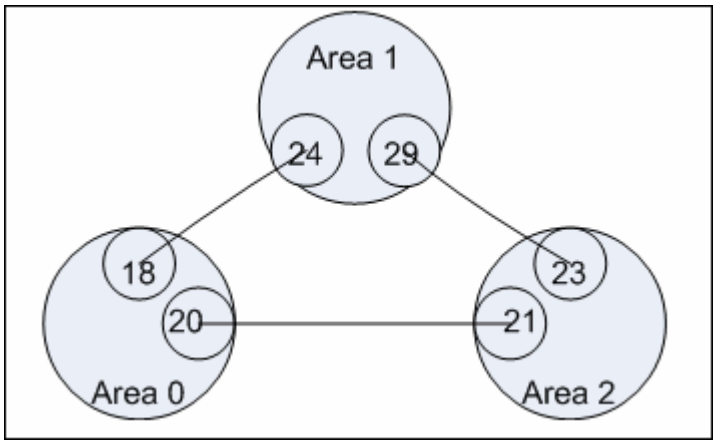

Figure 17: Hierarchical Model of the Network

Figure 17 shows the different levels in the network. The network is segmented in areas that contain border routers connecting them.

The global agent is the one that will make the first decision at the router. This agent will look at the area of the destination node. If the area is the same as the source node, then the packet is handled by the local agent. However, if the destination node belongs to an external area, it is the duty of the global agent to decide which border router in the local area the packet should be sent to. There will be two global Q-values that will be stored in each router in order to make these global decisions. The first one is used if the destination belongs to the local area. In this case the packet uses the regular Q value used in Q-Routing.

Figure 18 shows a flowchart of the routing decision process. The local decision has already been explained. The inter-area routing processing will be explained next.

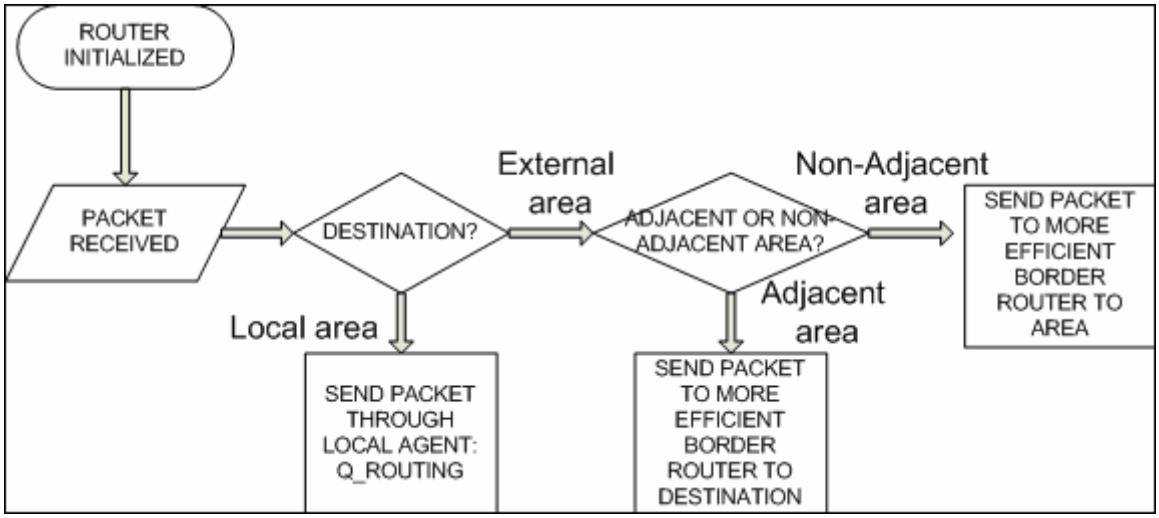

Figure 18: Proposed Method Flowchart

The second case takes place when a destination packet belongs to an area that is not adjacent to any of the border routers in the network. In this case each router will maintain a inter-area Q-value that has the following format:

Q_intearea[(source, destination area), border router]

Once this decision is made, the packet is encapsulated with a local destination that belongs to this border router. Once the packet reaches the border router, the border router discards the encapsulated destination and looks at the original destination. The border now must decide what border router belonging to the adjacent area is more efficient in order to deliver the packet to the destination area rapidly.

The main goal is to send the packet as quickly as possible to the destination area until it reaches an area that is adjacent to the destination area. At this point the adjacent area will be able to make a good decision to deliver the packet to the destination the most efficient way.

Adding a hierarchical management to deliver packets has numerous advantages. Q-Routing works well with smaller networks. The reason is that as the network topology grows larger it will take longer for values to be updated and therefore the adjustment to changes in the environment would be very slow. There is also the problem with the routing table. Routers in Q-Routing maintain a routing table with the best possible path using any neighbor to all destinations in the network. When the network is very large, every router must maintain Q-values to all destinations in the topology. This becomes very inefficient. In the new hierarchical model the border routers belonging to a specific area will contain the information about other areas in the topology and information will be updated quicker and more efficient than in the Q-Routing algorithm. SAHQ-Routing will be able to efficiently route packets in larger network topologies. This would also allow us to reduce the number of loops taking place in the network greatly.

6.4 Simulated Annealing based Q-Routing

The main goal of the Q-Routing problem is to be able to efficiently send packets from any source to any destination in the network. Thus it makes sense that if the network does not have a high load of packets traveling through it that it maintains the shortest path values since they are the optimal ones. The only way a problem would take place is when congestion takes place. The basic Q-Routing algorithm relies in the Q-values to adjust to the congested environment but it may not do so in the most optimal manner. Some of those Q-values may be out to date and therefore packets may be transmitted from further regions of the network into the congested area. In order to overcome this problem the proposed method implements a trigger in the border routers to advertise other routers in its area that it is starting to suffer from some congestion. The idea is not to solve the problem of a router already being congested but to prevent the router ahead of time of getting rapidly congested by choosing alternative paths. This is done by using simulated annealing based Q-routing using the waiting queue utilization of the routers as the temperature.

In order to be able to apply this concept efficiently in the routers, they must be able to support quality of service. A separate queue will be needed in order to manage these packets. This information is crucial to prevent the network from creating a bottleneck that will decrease the performance of the network at any time. Thus we will have the regular queue where network packets travel through and the management queue where packets are sent to rapidly adjust values in the network in order to avoid congestion. When a router receives a congestion management packet, it will look at the source address belonging to a border router in its area and will make the necessary

changes in the Router Congestion Table. This table contains the congestion utilization level of every border router in its area. Figure 19 shows how the queue management works inside of each one of the routers.

This QoS queuing method has not been implemented in the proposed algorithm. The proposed algorithm uses the First In First Out queuing method. Implementing this priority queuing method into the SAHQ-Routing would most likely improve the performance of the algorithm tremendously. It will be implemented in future work.

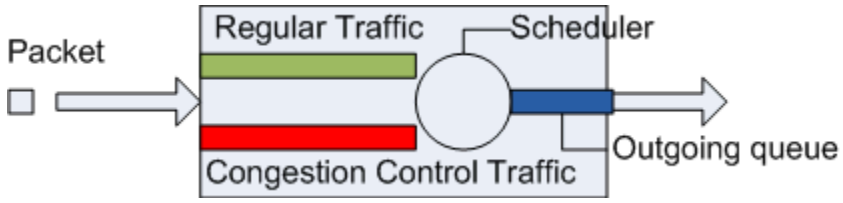

Figure 19: QoS Traffic Management

Border routers will start sending congestion management packets when their queue reaches 10% utilization.

In a previous section we looked at how simulated annealing was applied to Q-learning. Figure 16 indicated the following:

Select an action a_r in $A(s)$ arbitrarily
Select an action a_p in $A(s)$ according to the policy.
Generate a random number e 3 (0,1)
If $(e < exp((Q(s, a_r) - Q(s, a_p))/temperature)$ then $a \leftarrow a_r$

Figure 20: Simulated Annealing Decision Making

This states how exploration is performed. In the proposed method some changes will be applied. Exploration will only start taking place when needed. It will happen whenever a router starts experiencing some level of congestion. It will send the

congestion value to the other routers in the area and the Router Congestion Table will be updated in every one of these routers. This value is crucial since it represents the temperature. The other difference is that instead of choosing an arbitrarily action to explore, this method will explore the second best. The reason is that in an on-line scenario we do not want to be sending packets to areas that will retain the packet traveling through the network for long periods of time and therefore helping the congestion issue.

```
Select the second best action a_sec in A(s) according to the policy
Select an action a_p in A(s) according to the policy
Generate a random number e Є (0, 1)
If (e < exp ((Q(s, a_sec) – Q(s, a_p)) * temperature) then a ← a_sec
```

Figure 21: Implementation of Simulated Annealing in the proposed method

We want to have some level of exploration but it must be intelligent exploration if we want the routing algorithm to perform efficiently.

6.5 Algorithm Pseudo Code

In this section the pseudo code of the algorithm is described in detail. It is important to notice that Q* represents the optimal Q values, which is not to be confused with the arithmetic operation. For the routing problem Q* represents the shortest path from any node to any specific destination.

At initialization of the network all routers learn about the topology of the network and initialize their Q values to the shortest path to destinations.

There are two types of Q-values used in the computation of the proposed method. How these values work will be described from the moment a packet is created in the network until it reaches its final destination. The different scenarios will be explained in detail next.

When a packet is injected in the network the source router will decide whether the destination router is in the same area, an adjacent area or a distant area.

If the destination happens to be in the same area then the packet will be sent using Simulated Annealing based Q-Routing. The packet is sent to destination through a neighbor router following a policy and a reinforcement value is returned. The decision process is as follows:

Let x be the current node
Let d be the destination node
Select y' such that y' has minimal $Q_x(y', d)$ from all neighbor routers y of router x
Wait for a reply from y'
Update $Q_x(y', d)$ using the new estimate from router such that
$Q_x(y', d) = Q_x(y, d) + \alpha [Q_{y'}(z', d) + t + q - Q_x(y', d)]$ where $z \in$ neighbor routers of y'

Figure 22: Local area Decision Process

In the above representation, *x* represents the current node, *y* represents the neighbor router or the path to take and *d* represents the destination router. The variable *t* represents the transmission time and *q* represents the amount of time spent in the waiting queue. The reason why Q-routing is performed at a local level is because the values within the area will be updated very often and thus the Q-values will be very accurate to the ones taking place in real time. The main problem takes place when the network experiences a peak of high network load and goes back to low network load. Some of the Q-values will not be updated since the algorithm is purely greedy and therefore some values will maintain the highest Q-value even though they may not be the most optimal. In order to avoid this we keep track of the change in Congestion from one router to another. If the value is negative, then congestion is decreasing and therefore we must perform some exploration to make the Q-values more accurate. Since this takes place in a

local area, the packet will not take long to reach destination even if it takes the longest, more inefficient path.

If the destination happens to be in a different area then we will have to decide which border router in our area will be responsible for delivering the packet to the destination area. The main factor in this process is to figure out if there is congestion going to the destination area through any of the border routers. If the local routers contain this information they will be able to make an intelligent decision in order to avoid congestion.

If the destination happens to be in an adjacent area, then once the packet arrives to the border area it will use local Q-Routing to deliver the packet to its destination. Border routers contain information about areas that are directly connected to them so they can send a packet using regular Q-Routing.

The main purpose of the proposed method is to find an efficient way to detect congestion and to be able to propagate this information to other areas in the network so that exploration can be made avoiding the congested area. There are three types of packets that will be used to manage network congestion. One packet type is used to advertise congestion taking place in a border router of an area to the closest border router in each adjacent area. The other packet will be used to advertise congestion to the local border routers directly connected to the congested area by the previously mentioned border router already been advertised. The last packet will sent to all local routers in the area belonging to the congested router.

The process to advertise congestion from a border router is as follows:

```
Let B be the set of closest border routers in adjacent areas to the congested area and
all routers in the congested area
qw_x = waiting queue utilization at current border router x
If (Congestion Level() != qw_x)
        Send a congestion packet to B
```

Figure 23: Congestion Advertisement

There are two different types of congestion packet. Congestion packet type 2 is sent to the routers belonging to the same area as the congested router and the closest border routers belonging to adjacent areas to the congested area. Congestion packet type 3 is sent from the border routers belonging to adjacent areas that have received a congestion packet type 2 to other border routers in their areas. The process of receiving a congestion packet and updating the proper information is as follows:

```
Let q be the congested border router that sent the congested packet
Let qval be the congested value to be updated
Let x be the current node
Let A be the set of all areas in the topology
If (congestion packet type == 2)
    Congestion_x(q) = qval
    penalty = Q_interarea_x(q) * qval
    For every a Є A
        Q_intearea_x(q, a) = Q_interarea_x(q, a)* + penalty
    If (area(x) != area(q))
        Send a congestion packet type 3 to other border routers in the area
If (congestion packet type == 3)
    Congestion_x(q) = qval
    penalty = Q_interarea_x(q) * qval
    For every a Є A
        Q_intearea_x(q, a) = Q_interarea_x(q, a)* + penalty
```

Figure 24: Congestion Update Process

The third packet type is sent to all routers belonging to the same area as the congested router. Local routers will have this information and will be able to change the destination of a border router if this one is experiencing heavy congestion as opposed to

the others. The updates will be done the same way as the previous one but including only information about the local area.

When the congestion packet arrives to the closest border router connected to the congested area, it will update its congestion information and it will send another congestion packet to the other local border routers that are directly connected to the congested area. This packet will be a different type from the one sent by the congested border router. Figure 25 illustrates the process previously explained.

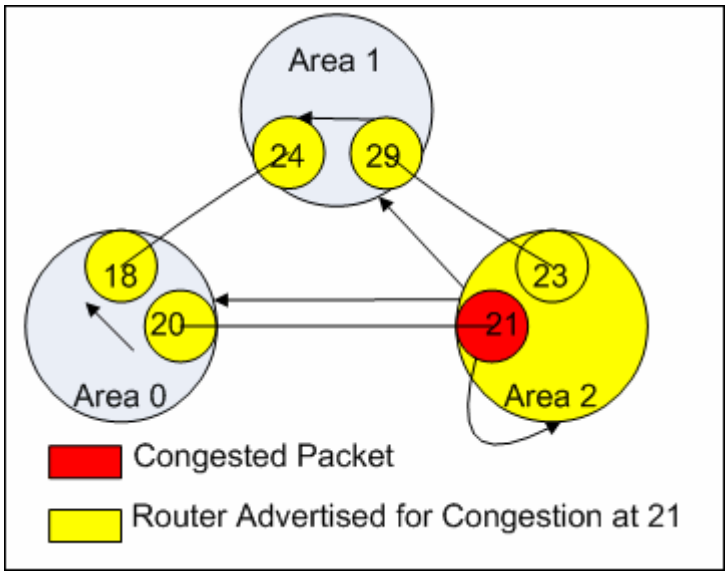

Figure 25: Congestion Advertisement Illustration

When a border router receives a regular packet, and the packet destination happens to be in an area other than the local or adjacent area, it must perform some computation in order to figure out whether to explore a new path or to send it though the optimal path. Since local router also receive a congestion packet if a border router in their area is experimenting network congestion, they will also be able to perform simulated annealing to select a different border router. In order to do so, the border router performs the following computations:

```
Let x be the current border node
Let A be the set of border routers in adjacent areas directly connected to x
Let d be the destination area
Select r' such that r' has minimal Q_interarea_x(r', d) from A
If (Congestion Level_x (r', d) != 0)
        Select r'' such that r'' has minimal Q_intearea_x(r'', d) where r'' ≠ r'
        Let B be the set of border routers in local area of x NOT including x
        Temperature = Congestion Level_x(r', d)
        Select a random number e where 0<= e <= 1
        If (e < EXP (Q_interarea_x*(r'', d) – Q_intearea_x*(r', d)) * temperature)
                Select r''
```

Figure 26: Inter-area border selection processing

The border router will first check to see the destination packet and its type. If the packet was sent to him and it is a congestion packet then the border router updates the congestion value. The idea is to avoid sending all packets from the local area to any destination that is attached to the border router that is experimenting congestion at this point.

The border router then checks to see its waiting queue utilization value. If it lies within one of the congestion levels then it must inform its adjacent area border routers about the congestion. The border router must avoid sending congestion packets every time it receives a new packet. To avoid flooding the network with constant congestion packets a congestion level range has been defined to update congestion values. This range has been arbitrarily chosen and different ranges will be used in order to see the different results.

Once the router has updated congestion values it is time to send the next packet in queue to its destination. The border router will check to see if the destination is adjacent or not. If the destination happens to be adjacent and since border routers keep information about both local and adjacent areas, it can send the packet using Q-Routing. If the

destination packet is further away, then we must see what adjacent router is best to send the packet to that area. This is when the router must check to see if that route is congested. It looks at the congestion value to send a packet to the destination area through a border router from an adjacent area. After the border router is chosen, the current router must make sure that no congestion is currently taking place going to the destination area through this adjacent area. If congestion is taking place then simulation annealing exploration will be used to load balance the network traffic.

CHAPTER 7

EXPERIMENT METHODOLOGY AND RESULTS

The proposed method has been implemented in the Java programming language. The code written by Littman and Boyan to develop the Q-Routing algorithm was used and translated to the Java language. Using this algorithm as a base, the necessary modifications were made to develop the proposed algorithm.

The experiment has been performed using the same 6x6 used in the original Q-Routing algorithm. This way the graphs will be able to show if the proposed method is able to perform better under the same environment.

The second experiment has been performed by slightly changing the 6x6 network topology. A new link was added to show better network traffic delivery by load balancing the traffic among different links.

The third experiment has been performed under a larger environment. This environment is also quite different since it offers redundancy between all areas in the topology. This last experiment will be included in Appendix A.

In the experiments that were conducted the following fixed ranges were used to represent the network loads:

Low Load: less than 1.5 (packets/time step)
Medium Load: 1.5 – 2.5 (packets/time step)
High Load: more than 2.5 (packets/time step)

The values in the ranges described above represent the average number of packets entering the network at a time step. It basically represents the rate at which the packets are being injected into the network.

7.1 Experiment Results in the 6x6 Network Topology

7.1.1 Medium and Low Network Loads

Medium network load ranges between 1.5 and 2.5 packets injected in the network at a time step. Both algorithms should perform very efficiently once the topology of the network has been learned.

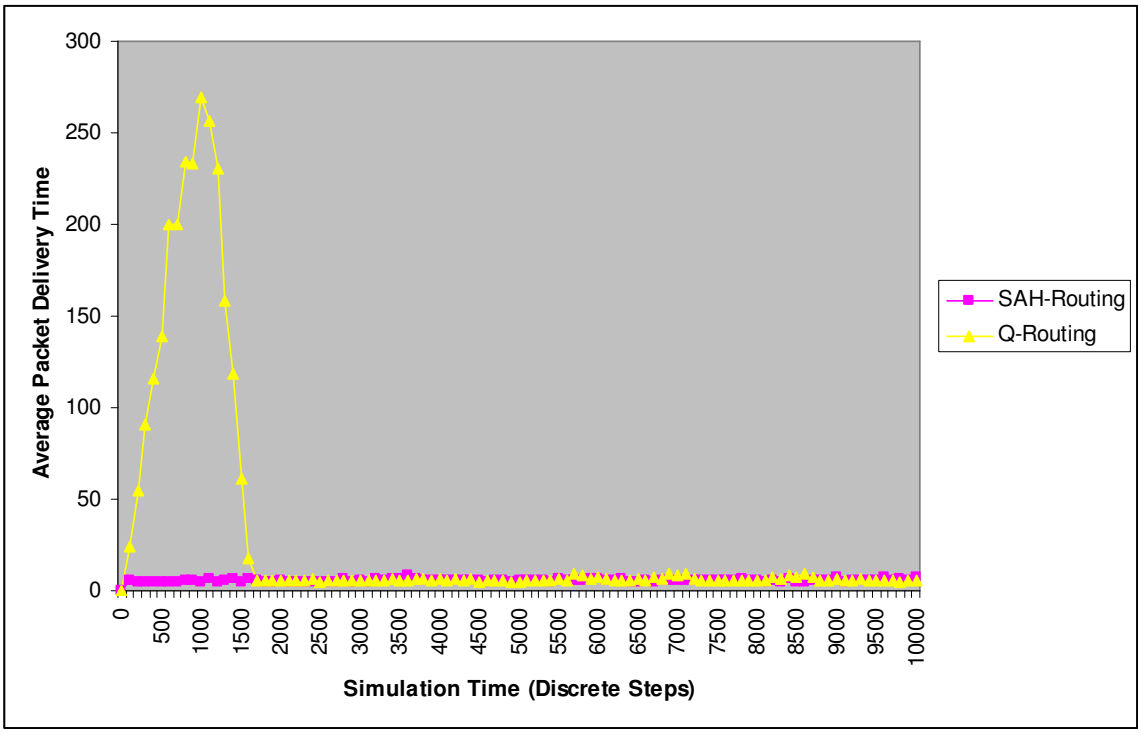

Figure 27: Low-Medium Network Load

SAHQ-Routing is able to load balance quick enough to maintain an efficient flow of traffic through the network. Q-Routing takes about 1700 time steps to converge and route packets efficiently across the network. Q-Routing is able to learn quicker about its

topology in medium load networks since there is more traffic flow and therefore is able to update its Q values quicker.

7.1.2 Dynamic Adaptation of Changing Loads

The main goal of any routing algorithm is to be able to adjust to the changing environment as quickly and as efficient as possible. In this section the proposed algorithm will be tested in changing network loads in the environment. As previously explained in previous chapters, Q-Routing is able to adjust to changing environments though the use of Q-learning. In this section Q-Routing and SAHQ-Routing will be compared in order to see which algorithm is able to adapt quicker to the changing environment.

7.1.2.1 Increasing network load

The first experiment that will be conducted under changing environments will be an increasing network load. In real life networks there are specific times of the day when most people utilize the network simultaneously creating large amounts of traffic flowing through the network.

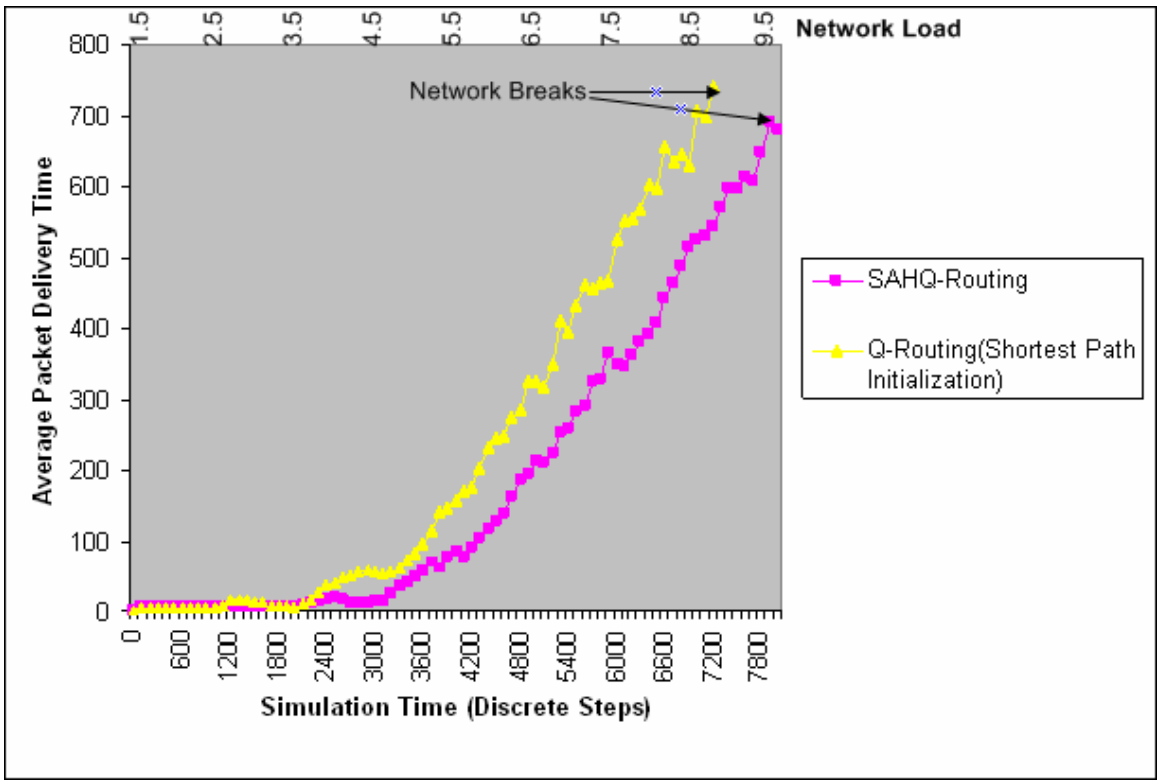

Figure 28: High Constant Network Load

Figure 28 shows the result of the experiment. It can be seen that under a rapid increase of network load SAHQ-Routing is able to balance the network faster than the Q-Routing algorithm. Q-Routing network breaks at time 7000. SAHQ-Routing is able to run until time 8000. Through the use of a more hierarchical network design and the use of simulated annealing the traffic is balanced quicker across the network. As the network load increases the difference in performance becomes more apparent.

7.1.2.2 Variable Network Load

This experiment is the one that represents a more realistic scenario in any network. There is usually a time in the day when people connect more to the network to check for emails, surf the internet, etc. There is a high increase of network load during these times and eventually the network load decreases.

The experiment starts with an average load of 0.5 (packets/time step). It then increases 1.0 every 1000 simulation steps. At time 5000 the network load falls back to 1.0 (packets/time step).

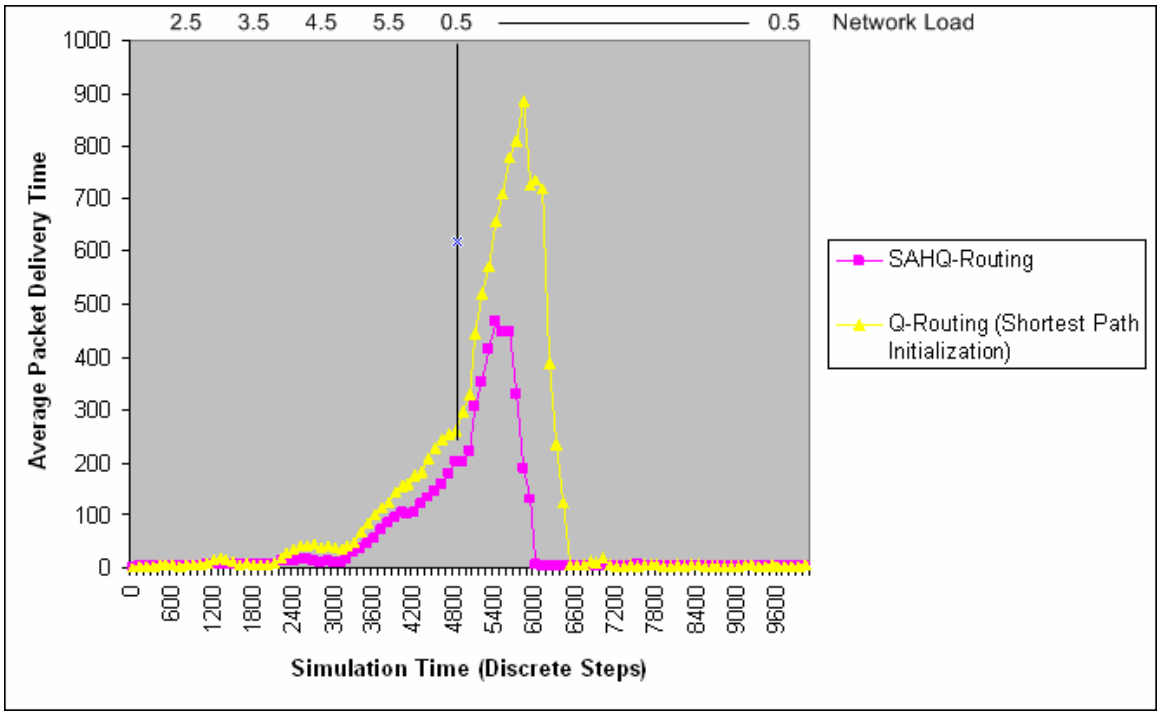

Figure 29: Variable Network Load

Figure 29 shows the results of the experiment. They show that not only SAHQ-Routing is able to route packets more efficiently under increasing network loads, but it is also able to recover from congestion faster. The reason why this happens is that Q-Routing relies on its Q-values to route packets efficiently. It learns a complete new set of values as Congestion increases. The problem is that by the time congestion ends, the Q-values are not optimized to work under low network loads and therefore it must start learning in order to adjust these values. During this time period the network already had many packets still being routed and since the Q-values are no longer adjusted to the new environment, packet will be routed in a non-efficient manner. Loops will be very frequent

during these time period. On the other hand, SAHQ-Routing works in a more hierarchical manner. It uses real time information to make decisions based on current network load. The simulated annealing congestion variable representing the temperature allows us to adjust inter-area traffic values rapidly to switch back from high to low congestion.

Congestion ends at simulation step 5000. SAHQ-Routing is able to realize this and make changes quickly to continue routing packets efficiently. Q-Routing must learn and updates its Q-values for some time until is able to route packets in an efficient manner. SAHQ-Routing is able to recover 600 simulation steps faster than Q-Routing. In a real case scenario this becomes very important since a routing protocol must be able to route packets efficiently under any circumstances.

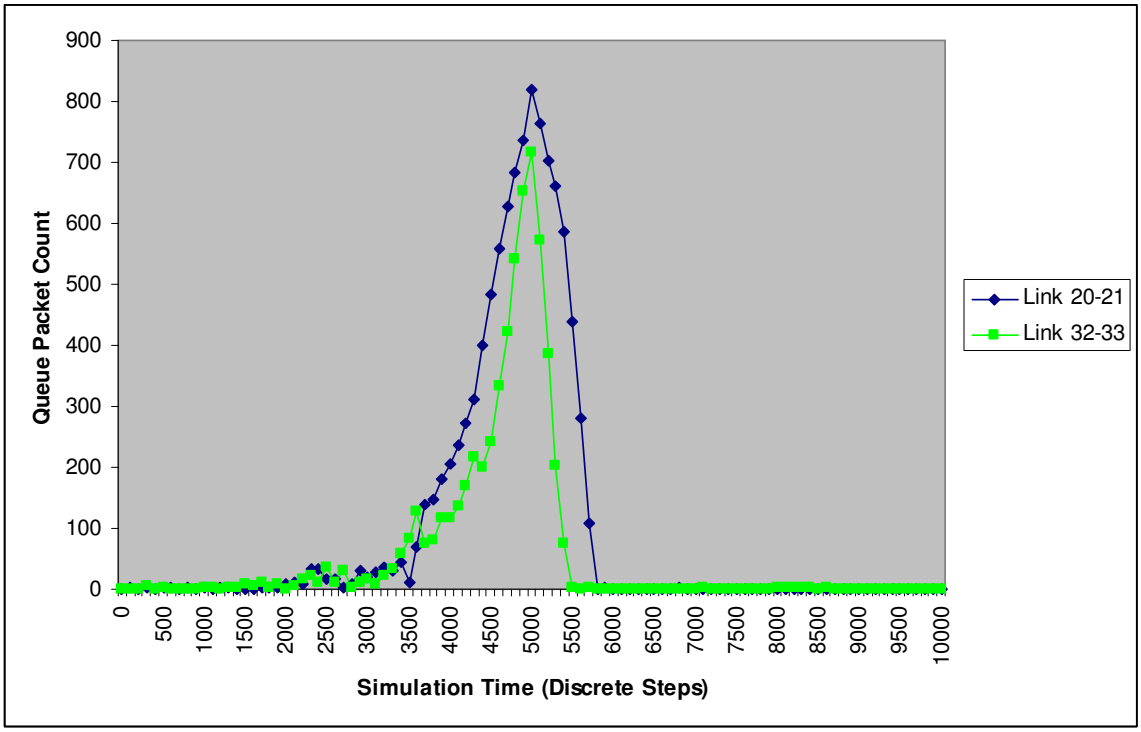

Figure 30: SAHQ-Routing Load Balancing

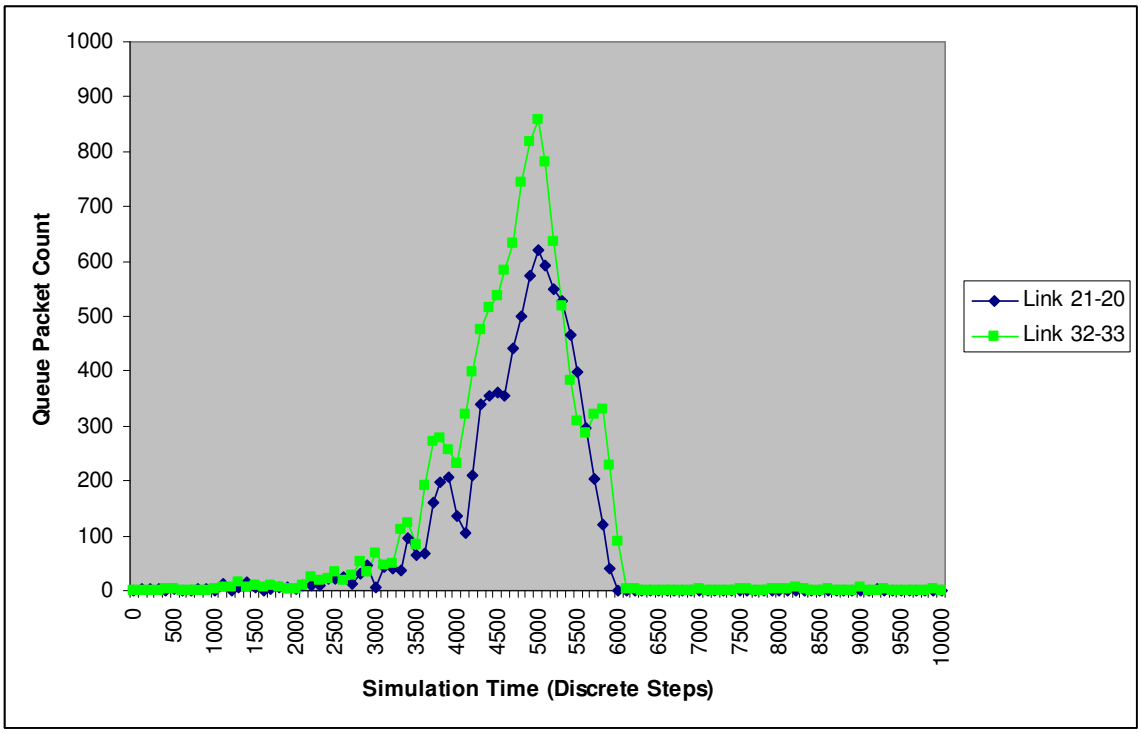

Figure 31: Q-Routing (Shortest Path Initialization) Load Balancing

Figure 30 and Figure 31 shows the load balance taking place in the network under the different routing algorithms. The data is taken from a single run just to show how traffic is being adjusted for load balancing across the network. The topology being used for this experiment is composed of two clusters connected by two links. Link 20-21 is the link that will yield the shortest path between most nodes from one cluster to the other. As the network increases the algorithm should notice this and start exploring alternative paths. In this case it should start utilizing Link 32-33 to alleviate congestion in Link 20-21. As we can see in the previous figure SAHQ-Routing succeeds at maintaining a balanced network load across these two links. Figure 31 shows how Q-Routing performs load balancing in the network through the use of its Q-values. SAHQ-Routing is able to keep the network load more balanced than Q-Routing does.

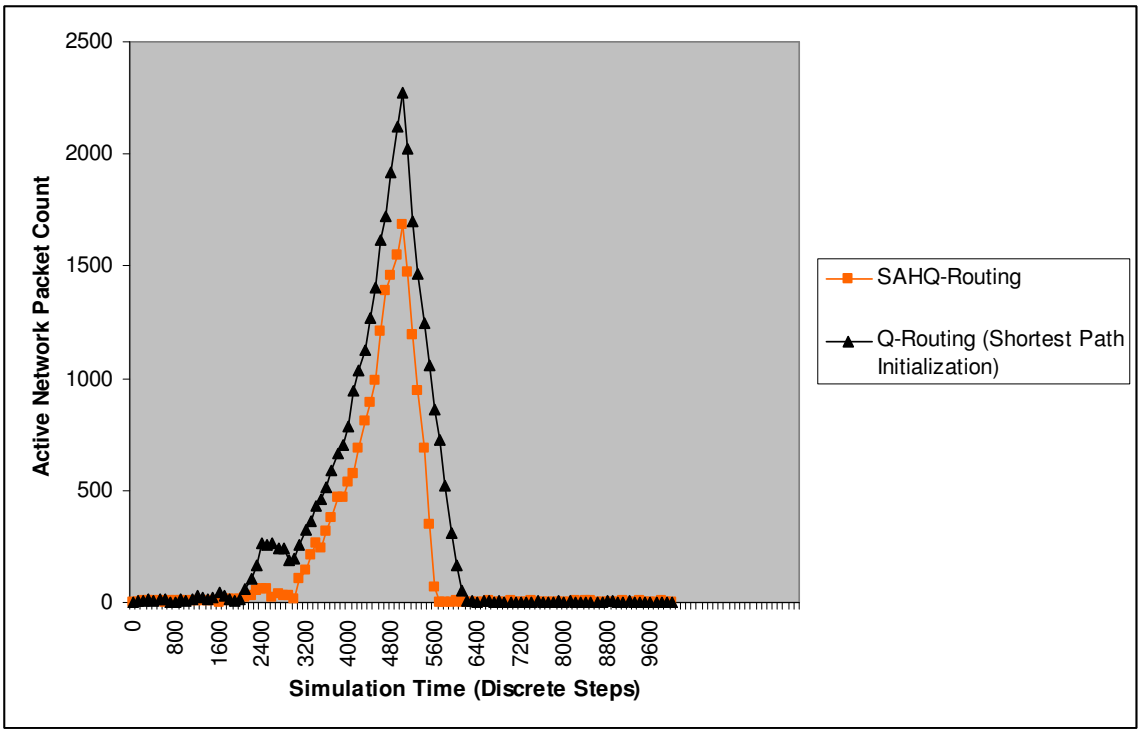

Figure 32: Active Network Packet Count

Figure 32 represents the number of active packets in the network at any given time. The results are also based on a single run. Since SAHQ-Routing is able to route packets more efficiently it is able to keep the network less congested than Q-Routing. Congestion ends at time step 5000. SAHQ-Routing adapts quickly to the changed environment and is able to route packets efficiently. We can see in the picture that packets are being routed quickly shortly after congestion ended. Since Q-Routing must learn about the changed environment it goes though a period of time after congestion ends where packets are not being routed efficiently and therefore the number of packets being routed in the network increases significantly.

7.2 Experiment Results in the Extended 6x6 Network Topology

In the previous experiment the network topology had two potential bottleneck links. The first link was the one connecting nodes 20 and 21. The second one was the link

connecting nodes 32 and 33. In a good network design, the designer must avoid having potential bottlenecks to avoid congestion. I have modified the previous 6X6 network by including a link connecting nodes 2 and 3.

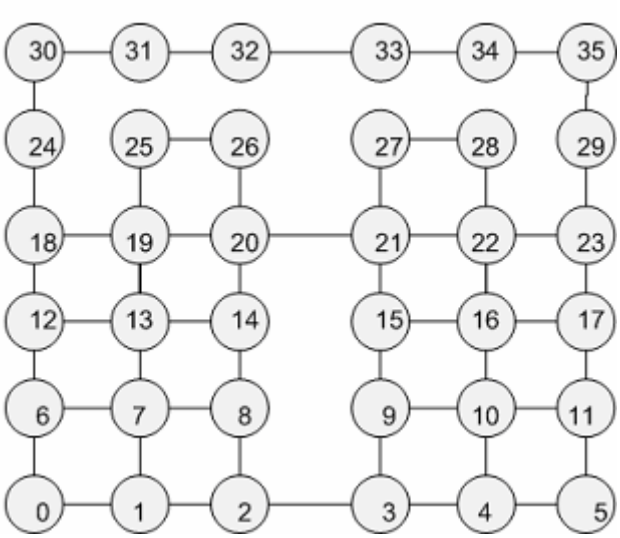

Figure 33: 6x6 Network Topology favoring load balancing

The new network topology favors load balancing for the proposed algorithm. The network has now three different paths connecting the east and west side of the network. There are three links available to distribute traffic evenly among these two areas. The next section will provide the results of the experiments conducted under this topology. The average packet delivery time should be lower in high congested network loads than those obtained in the previous network topology.

7.2.1 Variable Network Load

The following experiment has been conducted using the same parameters used in the previous 6x6 network topology under variable load networks. This environment scenario has been chosen since it is the most realistic one in real life networks and it allows us to see the behavior of the proposed algorithm under different network loads compared to that of the Q-Routing algorithm.

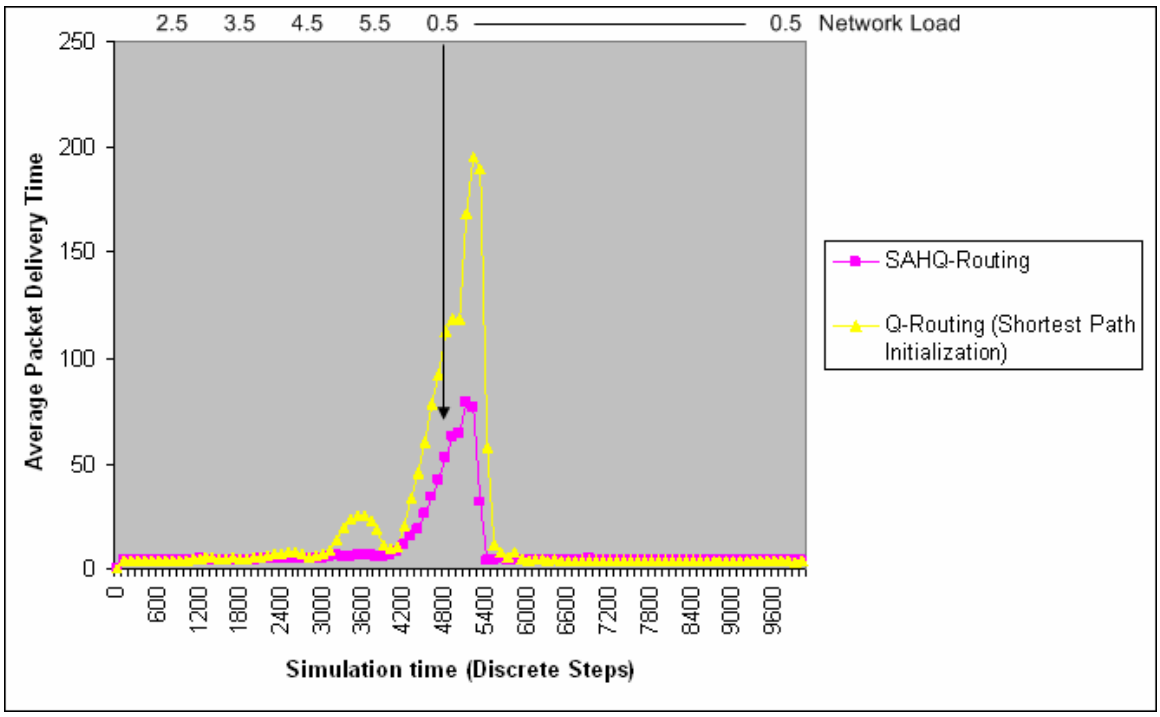

Figure 34: Packet Delivery Time in Extended 6x6 Grid Topology

In this experiment it can be noticed that the average delivery time is much lower than that of the previous experiment. Link 2-3 has been introduced to add a new path between the two main clusters. Load balancing is now active between three links instead of just two.

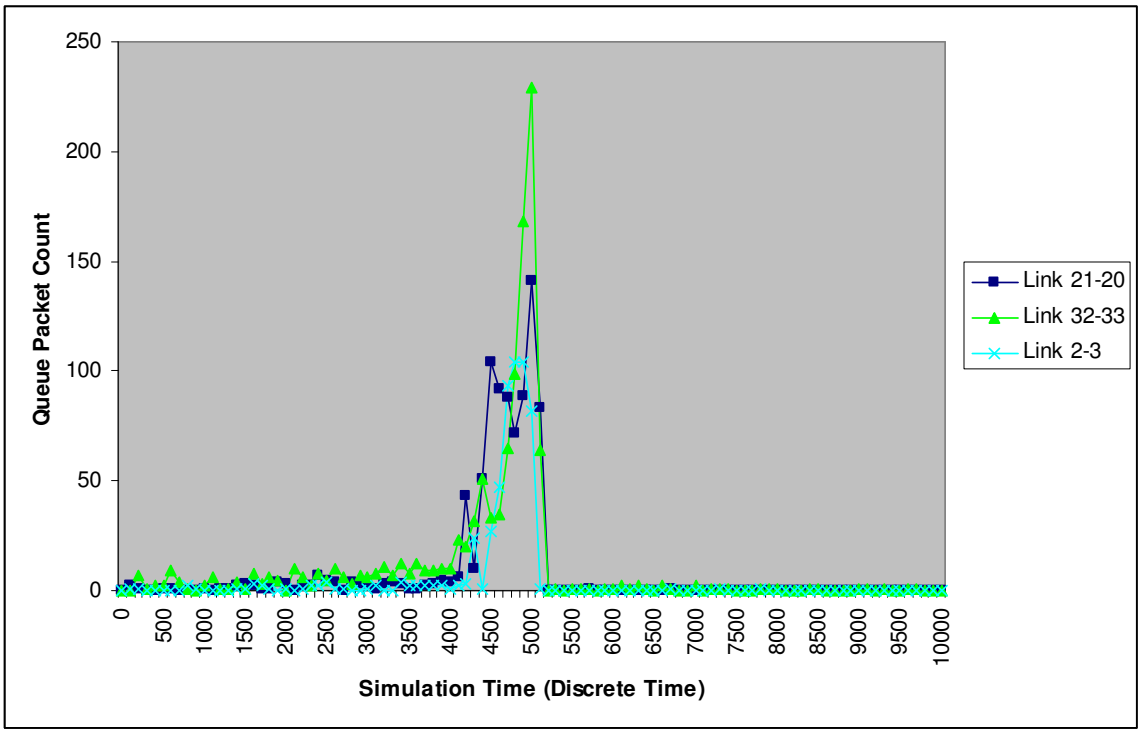

Figure 35: SAHQ-Routing Load Balancing in Extended 6x6 Grid Topology

Figure 35 shows how the packet traffic is balanced between the three links. During Congestion the three links are utilized in order to improve the routing. When one link starts experiencing a high peak, it can be notice that immediately other links start being utilized in order to balance the network load.

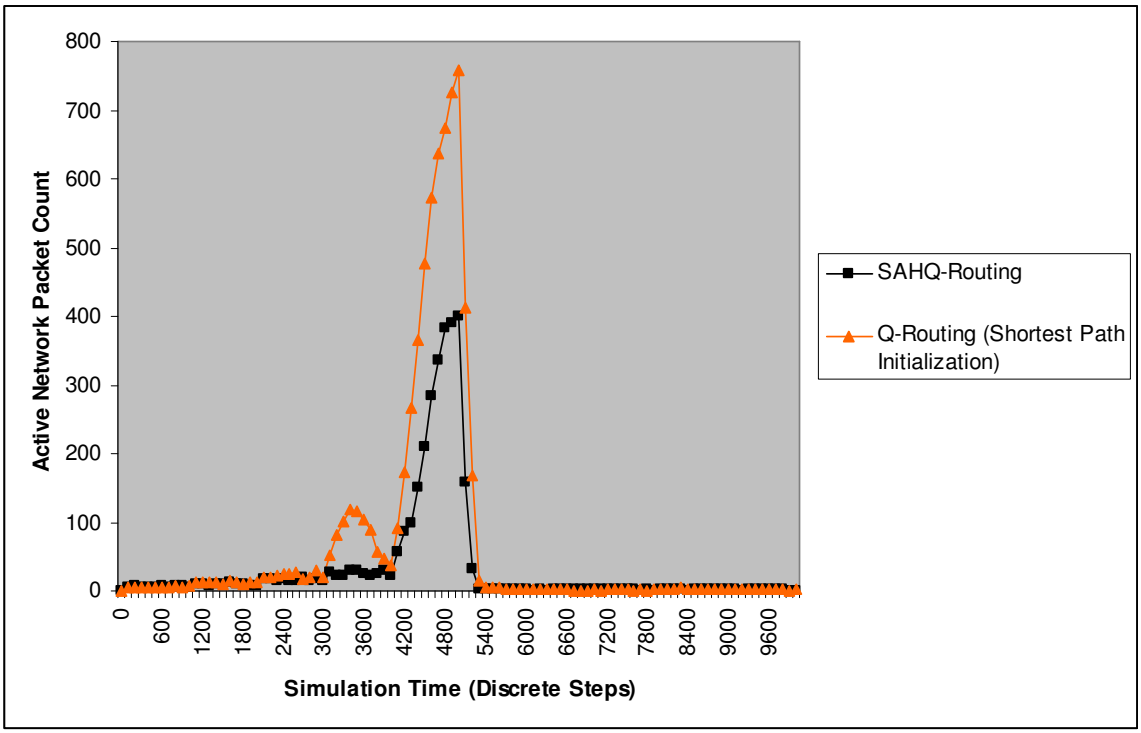

Figure 36: Active Network Packet Count

Figure 36 shows that at the highest peak of network load the number of packets in the network is higher when running SAHQ-Routing. The reason is that when congestion starts, SAHQ-Routing floods the local network with congestion packets to advertise their local nodes about the current congestion. The important fact is that with the help of these packets, routing is much more efficient and therefore the total routing time is much lower than that of Q-Routing. As shown in each experiment, Q-Routing must learn once the network falls out of congestion. Even though Q-Routing has fewer packets in the network, these are not being routed efficiently during this period of time.

CHAPTER 8

FUTURE WORK

The experiments conducted show that SAHQ-Routing is able to route packets more efficiently than Q-Routing in low, medium and high constant network loads. They also show that it performs better under variable network loads where it is able to adjust more rapidly than Q-Routing to congestion as well as recovering back to a low network load environment. For SAHQ-Routing to run efficiently is very important the way the areas are selected. We must be able to localize the possible links that could create potentially bottlenecks in the network and be able to apply areas accordingly to distribute the traffic between these links in times of congestion. At this point the areas are selected manually after analyzing the network topology. A great improvement of the algorithm is that each router would run a discovery mechanism at initialization and areas would be recomputed in order to assign areas to routers efficiently.

In order to make a more precise evaluation of the algorithm the simulator must be modified in order to use threading to represent each router. Instead of using time simulation as time steps, the experiment will have to run on real time. The distribution in which packets are introduced in the network will have to be stochastic.

Another important feature that is necessary for this algorithm to run efficiently is to apply priority queue management in the routers. For this experiment I had used First In First Out. It is evident that the algorithm must distinguish between normal traffic and the

traffic purely utilized for management purposes. This enhancement in the routers should make a noticeable difference in the performance of the algorithm in the network.

CHAPTER 9

CONCLUSION

The internet is increasing at a very rapidly rate. There is a need for routing optimization in order to speed up the way packets are delivered from hosts to destinations. There exist routing algorithms such as OSPF that use Shortest Path routing algorithm to perform the delivery of packets. These routing algorithms are efficient when the network load is not very high in the network but can become very inefficient when the network load increases. If the network load increases, the network becomes congested and bottlenecks begin to take place. The network then becomes inefficient. Littman and Boyan created the Q-Routing algorithm in 1994 so that the routers would be able to adjust to new changing environments dynamically. The Q-routing algorithm uses reinforcement learning to achieve this but still has some problems when adjusting to these new environments such as recovering from Congestion. A new method, Simulated Annealing based Hierarchical Q-Routing, is proposed in this thesis to overcome and improve the already existing Q-Routing algorithm.

The experiments conducted show that SAHQ-Routing is able to route packets more efficiently than Q-Routing in low, medium and high constant network loads. The reason is that Q-Routing must learn about the network before it is able to make efficient

routing. SAHQ-Routing is initialized to shortest path so that its values are optimal at initialization.

The most important part of the experiment was to see the difference in performance between SAHQ-Routing and Q-Routing under changing network loads. Two different topologies were used to see how both algorithms are able to adjust to changes in the environment. The first topology is the 6x6 grid used in the original Q-Routing simulations done by Boyan and Littman. The results of the simulation showed that SAHQ-Routing was able to maintain a more balanced traffic flow under changing environments and therefore the average queue routing time was lower than that of Q-Routing. Q-Routing is able to adjust to increasing network load by updating its Q-values. The problem with Q-Routing is that if the network rapidly suffers a change in the network load, Q-Routing takes some time to update its Q-values to the new environment and packets are not routed efficiently for a period of time. SAHQ-Routing uses a hierarchical model of the network to be able to advertise congestion to other areas so that congestion can be avoided. With a hierarchical model there more of a global view and we can exchange information between the highest layer of the network in order to avoid congestion. This way information can be propagated across the network much quicker and decision can be made using values very closed to the real ones.

New features can be added to the algorithm to enhance its performance. The addition of a discovery algorithm at each router's initialization would make this process dynamic without having to change information manually each time. The addition of priority queuing would definitely speed up propagation of management information across the network. In the current algorithm management packets are treated the same

way as regular traffic. This means that in times of congestion these packets may take some time to arrive to their destination and by that time something else may have changed in the network.

REFERENCES

[1] A. Mellouk, S. Hoceïni, S. Larynouna. “Flow based Routing for Irregular Traffic using Reinforcement Learning Approach in Dynamic Networks” In Proceedings of the 11th IEEE Symposium on Computers and Communications (ISCC’06), 2006.

[2] A. Mellouk, S. Hoceïni, S. Larynouna, “Adaptive Probabilistic Routing Schemes for Real Time Traffic in High Speed Dynamic Networks”, in IJCSNS International Journal of Computer Science and Network Security, Vol. 6 No 5B, pages 36-42, May 2006.

[3] M. Guo, Y. Liu, and J. Malec, “A new Q-Learning Algorithm Based on the Metropolis Criterion”, In IEEE transactions on Systems, man, and Cybernetics-Part B: Cybernetics, Vol. 34, No. 5, Pages 2140-2143, October 2004.

[4] S. Whiteson, and P. Stone, “Towards Autonomic Computing: Adaptive Network Routing and Scheduling”, In Proceedings of the International Conference on Autonomic Computing (ICAC’04), 2004.

[5] J. Boyan and M. L. Littman, “Packet Routing in Dynamically Changing Networks: A Reinforcement Learning Approach.” In Cowan, Tesauro and Alspector .(eds), Advances in Neural Information Processing Systems 6, 1994.

[6] J. Boyan and M. Littman, “A Distributed Reinforcement Learning Scheme for Network Routing.”, Technical report, Department of Computer Science, Carnegie Mellon University, 1993.

[7] R. S. Sutton and A. G. Barto, “Reinforcement Learning: An Introduction.” MIT Press, Cambridge, MA, 1998.

[8] S. Kumar and R. Miikkualainen, “Confidence-based Q-routing: an on-queue adaptive routing algorithm” In Proceedings of Neural Networks in Engineering, 1998.

[9] M.Abramson, and H. Wechler, “Tabu Search Exploration for On-Policy Reinforcement Learning”, Proceedings of the International Joint Conference on Neural Networks, Vol. 4, Pages 2910 – 2915, 2003.

[10] A. Nowe, K. Steenhaut, M. Fakir, and K. Verbeeck. “Q-learning for Adaptive, Load Based Routing”, 1998 IEEE International Conference on Systems, Man, and Cybernetics, Vol. 4, Pages 3965 – 3970, 1998.

[11] C. Yang. “Reinforcement Learning in a Car Simulator”, Technical report. Department of Computer Science, University of British Colombia, 2003.

[12] L.C. Baird, “Reinforcement Learning in continuous time: advantage updating”, IEEE World Congress on Computational Intelligence, volume 4, pages 2448 – 2453, 1994.

[13] M. Ricordeau, “Q-Concept-Learning: Generalization with Concept Lattice Representation in Reinforcement Learning”, In Proceedings of the 15th IEEE International Conference on Tools with Artificial Intellicence (ICTAI’03), 2003.

[14] S. Choi, and D. Yeung, “Predictive Q-Routing: A memory-based Reinforcement Learning approach to Adaptive Traffic Control.” Department of Computer Science, Hong Kong university of Science and Technology, 1996.

[15] Shailesh, Kumar, “Confidence based Dual Reinforcement Q-Routing: an On-line Adaptive Network Algorithm”. Master Thesis. The University of Texas at Austin, 1998.

[16] Coggan, Melanie, “Exploration and Exploitation in Reinforcement Learning”. Technical report. CRA-W DMP Project at McGill University, 2004.

[17] A. Nowe, K. Steenhaut, M. Fakir, and k. Verbeeck, “Q-learning for adaptive load based routing”, In IEEE transactions on Systems, Man, and Cybernetics, Vol. 4, Pages 3965-3970, October 1998.

[18] X. Jing, C. Liu, and X. Sun, “Artificial Cognitive BP-CT Ant Routing Algorithm”, in Proceedings of International Joint Conference on Neural Networks, Montreal, Canada, July 31 – August 4, 2005.

[19] P.R.J Tillotson, Q. H. Wu, and P.M Hughes, “Multi-Agent Learning for Control of Internet Traffic Routing”, Appears in Learning Systems for Control, IEE Seminar, 2000.

[20] Y. Achbany, F Fouss, L. Yen, A. Pirotte, and M. Saerens, “Managing the Exploration/Exploitation Trade Off in Reinforcement Learning”, Technical Paper, Information Systems Research Unit (ISYS), IAG, Université Catholique de Louvain, June, 2005.

[21] B. Srinivasan, “Robot Navigation in Partially Observable Domains using Hierarchical Memory-Based Reinforcement Learning”, In the 2nd International Conference on Ubiquitous Robots and Ambient Intelligence, 2005.

[22] N. Lilith, and K. Doğançay,”Distributed Reduced-State SARSA Algorithm for Dynamic Channel Allocation in Cellular Networks Featuring Traffic Mobility.” In IEEE International Conference on Communications, Vol. 2, 2005.

[23] M. Abramson, P. Pachowicz, and H. Wechsler, “Competitive reinforcement learning in continuous control tasks”, In Proceedings of the International Joint Conference on Neural Networks, Vol. 3, 2003.

[24] S. Khodayari, and M.J. Yazdanpanah, “Network Routing Based on Reinforcement Learning In Dynamically Changing Networks”, In Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence (ICTAI ’05), 2005.

APPENDIX A

The topology used in this experiment is the most logical when analyzing network topologies. Each area of the topology can get to any other area in the topology though various paths. This way if a link goes down or is congested the traffic can use another path to send the packet to its destination. Figure 37 shows the network topology that will be used in this experiment. Each square with a different color represents an area in the topology.

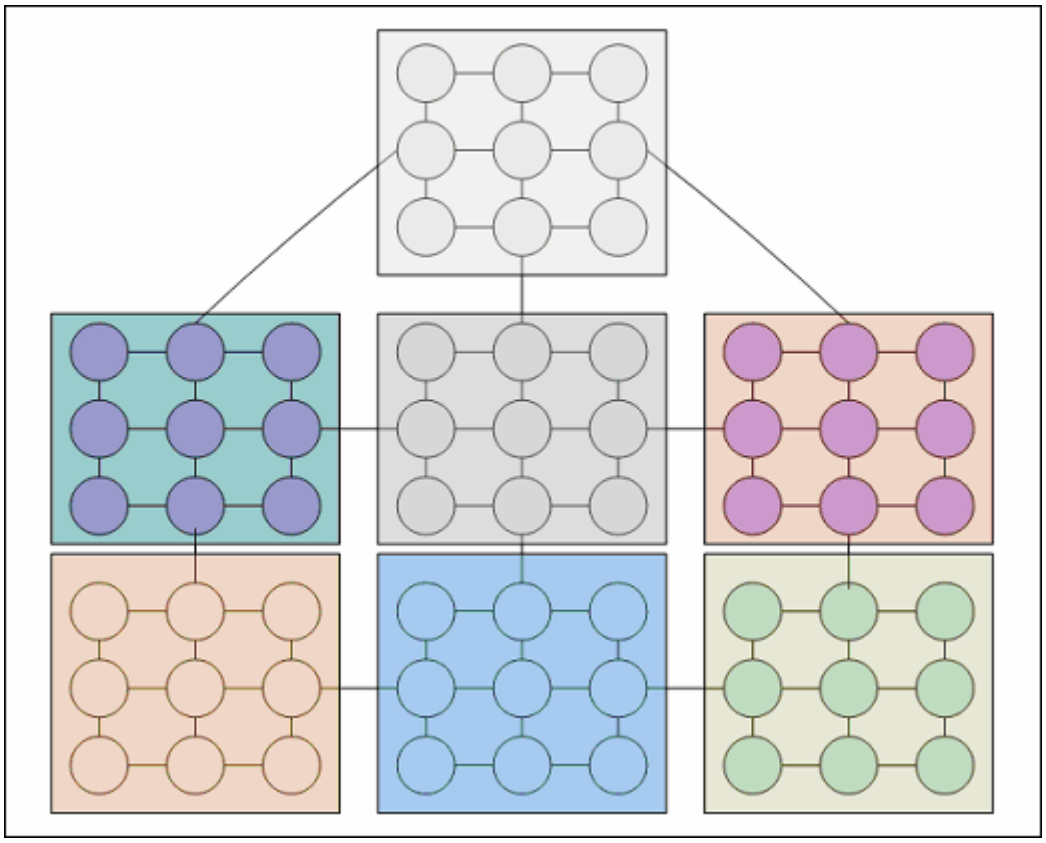

Figure 37: Redundant Network Topology

It is expected that both algorithms should work efficiently under this type of topology. However the proposed algorithm, simulated annealing based hierarchical Q-Routing, should adjust faster than the original Q-Routing due to the hierarchical packet delivery management.

This experiment is performed under variable network loads. In the previous experiments two clusters were connected by two and three links. As the network load increased, new paths less congested had to be found to balance the traffic of the network. In this experiment, due to the redundant design of the topology the traffic across the network should stay balanced at higher network loads than the previous experiments.

SAHQ-Routing should perform much better than Q-Routing with larger topologies. The reason is because of the hierarchical management to deliver packets. There is a global view of the topology and therefore SAHQ-Routing is able to adjust quicker and is able to make better routing decisions at all times. Q-Routing takes longer to learn about larger topologies. Q-Routing decreases in routing performance as the network topology becomes larger. Q-Routing uses Q-values that are maintained in the routers to make routing decisions. These estimates take longer to be updated and therefore routing becomes inefficient.

The type of topology used in this experiment also makes SAHQ-Routing perform much more efficiently than Q-Routing does. This is once again because of the hierarchical management of delivering packets. There is a lot of redundancy in the network and therefore there are multiple paths to avoid congestion. The sooner routers realize, the sooner congestion will be able to be avoided. As mentioned earlier, Q-Routing loses efficiency as network topologies grow larger. SAHQ-Routing due its

hierarchical functionality is able to adjust much quicker under changing network environments. Figure 38 shows the results of the experiment under various network loads. The difference in performance is overwhelming. Q-Routing has a difficult time adjusting to the changing network load and reaches a high peak of over 2000 average packet delivery time. SAHQ-Routing due to its preventive nature to avoid congestion is able to maintain a stable network flow and reaches a high peak of less than 500 average packet delivery time.

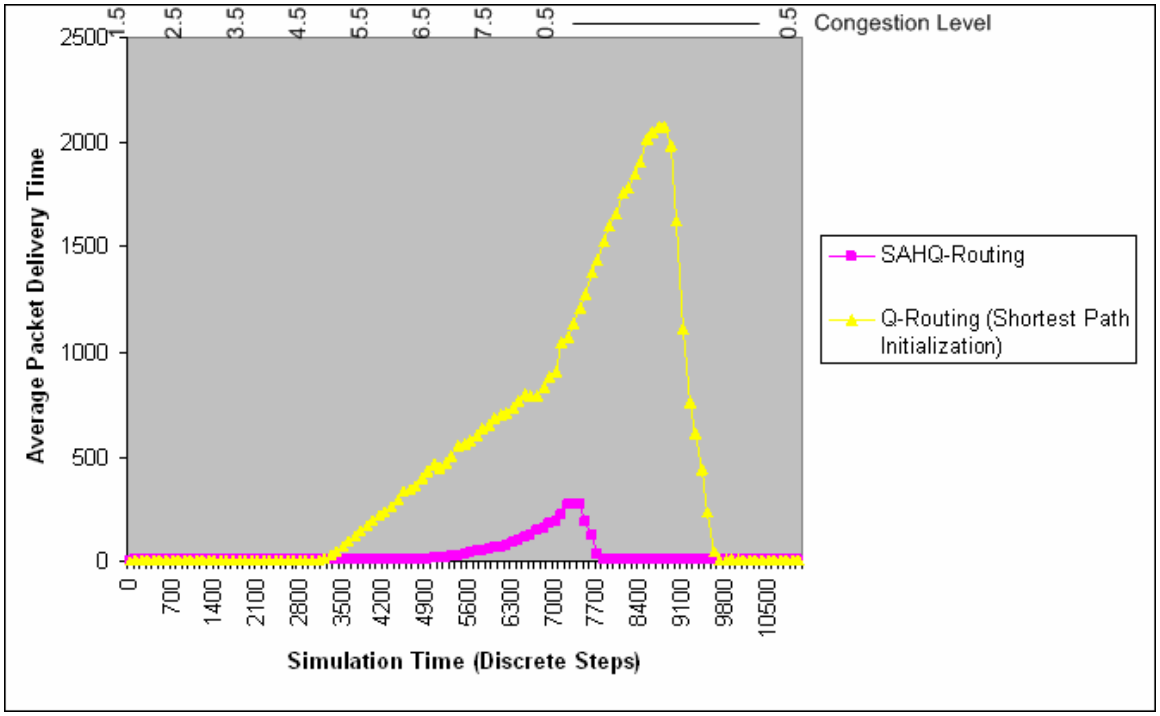

Figure 38: Average Packet Delivery Time

Figure 39 shows the active network packet count. SAH-Q-Routing reaches a maximum packet network count of about 2000. Q-Routing reaches a maximum network count of over 8000 packets. This shows how much more efficient SAHQ-Routing performs under this type of scenario. As the networks grow larger SAHQ-Routing is still

able to perform efficiently. Q-Routing decreases its functionality greatly as network topologies grow larger.

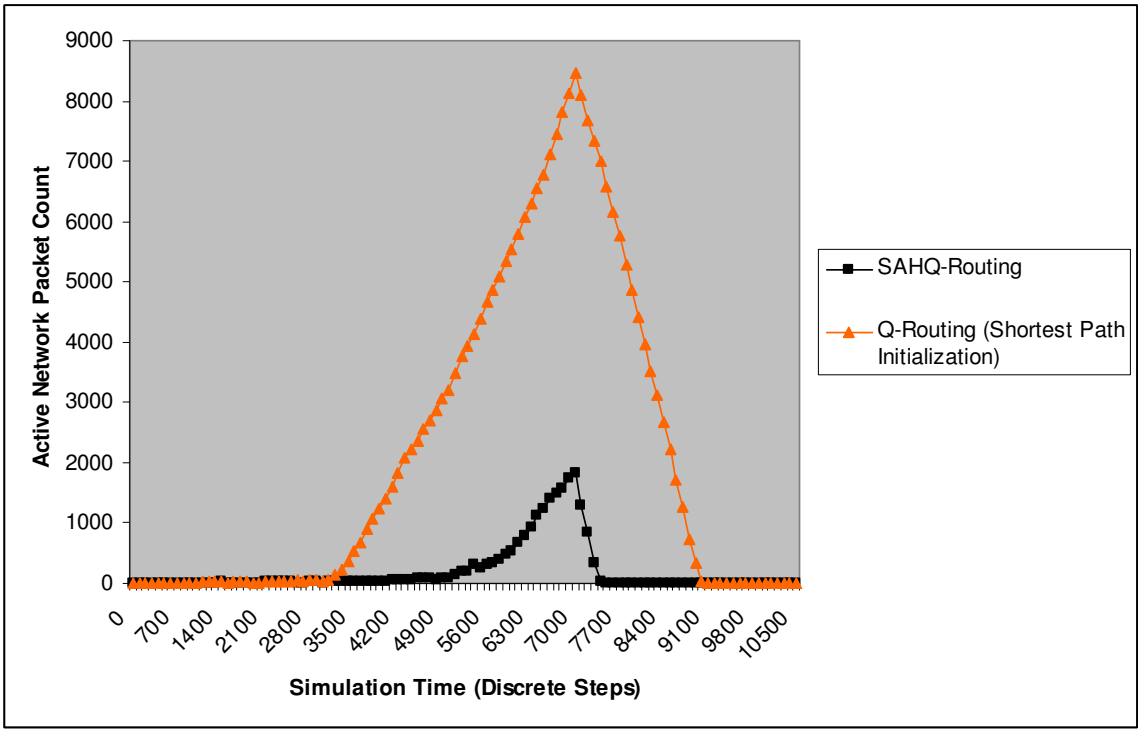

Figure 39: Active Network Packet Count

VITA

Antonio Mira Lopez

Candidate for the Degree of

Master of Science

Thesis: SIMULATED ANNEALING BASED HIERARCHICAL Q-ROUTING: A DYNAMIC ROUTING PROTOCOL

Major Field: Computer Science

Biographical:

Education:

- Completed the requirements for the Master of Science in Computer Science at Oklahoma State University, Stillwater, Oklahoma in December, 2007.
- Bachelor of Science in Computer Science at the University of Science and Arts of Oklahoma, Chickasha, Oklahoma in May, 2000.

Experience:

- Gestion Tributaria Territorial. Database and Web Programmer. Alicante, Spain. October 2000 – April 2003
- Oklahoma State University. Department of Electrical Engineering. Research assistant. Stillwater, OK February 2004 – February 2005
- Oklahoma State University. IT Telecommunications. Network Operations Center Technician. Stillwater, OK. May 2005 – Present

Name: Antonio Mira Lopez Date of Degree: December, 2007

Institution: Oklahoma State University Location: Stillwater, Oklahoma

Title of Study: SIMULATED ANNEALING BASED HIERARCHICAL Q-ROUTING: A DYNAMIC ROUTING PROTOCOL

Pages in Study: 68 Candidate for the Degree of Master of Science

Major Field: Computer Science

Scope and Method of Study:

The area of the research is artificial intelligence and networking. The method of study has been implementation and simulation of the proposed routing algorithm using the programing language Java.

Findings and Conclusions:

The focus of this paper is to provide an efficient solution to the routing problem by making use of reinforcement learning and other heuristics. The thesis describes a routing algorithm called Q-Routing based on the Q-learning algorithm that will be used as the base for the proposed Simulated Annealing based Hierarchical Q-Routing approach. The proposed approach adds functionality layers to provide a more hierarchical view of the network. Providing hierarchy by defining network areas and roles to routers within these areas allows us to have more updated global information and therefore much better decision making when routing packets across the network. The addition of simulated annealing as an exploration method also plays an important role in the improvement of the original Q-Routing approach. The results from every experiment that was performed show SAHQ-Routing to be a much more robust and efficient routing algorithm than Q-Routing.

ADVISER'S APPROVAL: Dr. Douglas K. Heisterkamp