Volume 3 ▪ Issue 1 ▪ March 2011

Editor-in-Chief
Dr. Min Song

INTERNATIONAL JOURNAL OF

COMPUTER NETWORKS (IJCN)

ISSN : 1985-4129

Publication Frequency: 6 Issues / Year

CSC PUBLISHERS
http://www.cscjournals.org

INTERNATIONAL JOURNAL OF COMPUTER NETWORKS (IJCN)

VOLUME 3, ISSUE 1, 2011

EDITED BY
DR. NABEEL TAHIR

ISSN (Online): 1985-4129

International Journal of Computer Networks (IJCN) is published both in traditional paper form and in Internet. This journal is published at the website http://www.cscjournals.org, maintained by Computer Science Journals (CSC Journals), Malaysia.

IJCN Journal is a part of CSC Publishers
Computer Science Journals
http://www.cscjournals.org

INTERNATIONAL JOURNAL OF COMPUTER NETWORKS (IJCN)

Book: Volume 3, Issue 1, March 2011
Publishing Date: 04-04-2011
ISSN (Online): 1985-4129

IJCN Journal is a part of CSC Publishers
http://www.cscjournals.org

Published in Malaysia

Typesetting: Camera-ready by author, data conversation by CSC Publishing Services – CSC Journals, Malaysia

CSC Publishers, 2011

EDITORIAL PREFACE

The International Journal of Computer Networks (IJCN) is an effective medium to interchange high quality theoretical and applied research in the field of computer networks from theoretical research to application development. This is the third issue of volume second of IJCN. The Journal is published bi-monthly, with papers being peer reviewed to high international standards. IJCN emphasizes on efficient and effective image technologies, and provides a central for a deeper understanding in the discipline by encouraging the quantitative comparison and performance evaluation of the emerging components of computer networks. Some of the important topics are ad-hoc wireless networks, congestion and flow control, cooperative networks, delay tolerant networks, mobile satellite networks, multicast and broadcast networks, multimedia networks, network architectures and protocols etc.

The initial efforts helped to shape the editorial policy and to sharpen the focus of the journal. Starting with volume 3, 2011, IJCN appears in more focused issues. Besides normal publications, IJCN intend to organized special issues on more focused topics. Each special issue will have a designated editor (editors) – either member of the editorial board or another recognized specialist in the respective field.

IJCN give an opportunity to scientists, researchers, engineers and vendors to share the ideas, identify problems, investigate relevant issues, share common interests, explore new approaches, and initiate possible collaborative research and system development. This journal is helpful for the researchers and R&D engineers, scientists all those persons who are involve in computer networks in any shape.

Highly professional scholars give their efforts, valuable time, expertise and motivation to IJCN as Editorial board members. All submissions are evaluated by the International Editorial Board. The International Editorial Board ensures that significant developments in computer networks from around the world are reflected in the IJCN publications.

IJCN editors understand that how much it is important for authors and researchers to have their work published with a minimum delay after submission of their papers. They also strongly believe that the direct communication between the editors and authors are important for the welfare, quality and wellbeing of the journal and its readers. Therefore, all activities from paper submission to paper publication are controlled through electronic systems that include electronic submission, editorial panel and review system that ensures rapid decision with least delays in the publication processes.

To build its international reputation, we are disseminating the publication information through Google Books, Google Scholar, Directory of Open Access Journals (DOAJ), Open J Gate, ScientificCommons, Docstoc and many more. Our International Editors are working on establishing ISI listing and a good impact factor for IJCN. We would like to remind you that the success of our journal depends directly on the number of quality articles submitted for review. Accordingly, we would like to request your participation by submitting quality manuscripts for review and encouraging your colleagues to submit quality manuscripts for review. One of the great benefits we can provide to our prospective authors is the mentoring nature of our review process. IJCN provides authors with high quality, helpful reviews that are shaped to assist authors in improving their manuscripts.

Editorial Board Members
International Journal of Computer Networks (IJCN)

EDITORIAL BOARD

TABLE OF CONTENTS

Volume 3, Issue 1, March 2011

Pages

1 - 9 On the Tree Construction of Multi hop Wireless Mesh Networks with Evolutionary Algorithms
Zahra Zareei, Shahram Jafari

10 - 16 An Algorithm for Computing Average Packet Delay
M. R. Hassan

17 - 36 Syed Muhammad Atif
Avninder Gill

37 - 42 Fuzzy Controller Based Stable Routes with Lifetime Prediction in MANETs
Taqwa Odey , Abduladhem Ali

On the Tree Construction of Multi Hop Wireless Mesh Networks With Evolutionary Algorithms

Zahra Zarei zahra.zarei@gmail.com
Department of Computer Science and engineering
Shiraz University , Shiraz, Iran

Shahram Jafari jafaris@shirazu.ac.ir
Department of Computer Science and engineering
Shiraz University , Shiraz, Iran

Abstract

In this paper, we study the structure of WiMAX mesh networks and the influence of tree structure on the performance of the network. From a given network graph, we search for trees, which fulfill some network, QoS requirements. Since the searching space is very huge, we use genetic algorithm in order to find solution in acceptable time.
We use NetKey representation which is an unbiased representation with high locality, and due to high locality we expect standard genetic operators like n-point cross over and mutation work properly and there is no need for problem specific operators. This encoding belongs to class of weighted encoding family. In contrast to other representation such as characteristics vector encoding which can only indicate whether a link is established or not, weighted encodings use weights for genotype and can thus encode the importance of links. Moreover, by using proper fitness function we can search for any desired QoS constraint in the network.

Keywords: Wireless Mesh Networks, WiMAX, Network Planning, Multihop Networks.

1. INTRODUCTION

Wireless mesh networks have received much attention in recent years due to its low up-front cost, easy network maintenance, robustness, and reliable service coverage [3-5]. Different from traditional wireless networks, WMN is dynamically self-organized and self-configured. In other words, the nodes in the mesh network automatically establish and maintain network connectivity.
In such networks, each mesh node plays both roles of a host and a router. Packets are forwarded in a multihop fashion to and from the gateway (connected to the Internet). It has been shown that the throughput and delay performances in wireless mesh networks are location dependent [6-9]. Mesh networks show great advantages such as good coverage, rapid and cost-efficient deployment, and robustness. Mesh networks are built using various technologies, however the most commonly used are WiFi (based on the IEEE 802.11 family) and WiMAX (based on IEEE 802.16). Using IEEE 802.11 for the wireless backbone leads to dense and suboptimal deployments due to the short transmission range of the standard and, consequently, low aggregate throughput capacity can be obtained. WiMAX, on the other hand, has a transmission range of several kilometers with high data rates.
In WiMAX mesh networks, a MSS (Mesh Subscriber Station) can only have one path to the BS (Base Station) which is through its immediate parent [1]. Thus, MSS nodes are organized in a tree structure rooted at the BS. The tree topology is constructed through temporarily disconnection of some links logically. From the other side, we know for a given network graph, one can construct plenty of spanning trees (see the Kirchhoff's Matrix Tree Theorem [2]). In each resulted tree, nodes fan-out and the tree's depth affected the performance of the WiMAX network markedly. In this paper, we investigate this phenomenon and find desired tree topologies, which satisfy intended delay and throughput trade-off. We first obtain per-node throughput and delay of each node using the model proposed in [9]. In this model, we can obtain throughput and delay for

each node independent of tree structure. Using the obtained results, and, employing a proper scheduling algorithm, we search for the best tree topologies. For this purpose, we use an evolutionary algorithm in order to find trees, which satisfy some QoS requirements. The proposed algorithm converges fast while it finds good enough answers. Due to huge searching space, classical searching algorithms lead to unacceptable searching time and thus are not practical.
The rest of this paper is organized as follows: in section 2, we discuss the related work. In section 3, we define model assumptions. In section 4, we present the genetic algorithm for tree construction. In section 5, we show the results. Finally, the paper is concluded in section 6.

2. RELATED WORKS

Designing mesh networks and specially WiMAX has been studied in some works.
In [4] the authors proposed an algorithm that selects a parent for each MSS, which maximize throughput capacity. The object is to select links that have highest data rates among the set of all possible paths between an MSS and the BS. In [9] the authors model and analyze the location-dependent throughput and delay in WMN. They analyze the packet arrival and the packet departure rate for the forwarding queues at each node, and based on the analytical model, they proposed two network design strategies to provide fair resource sharing and minimize the end-to-end delay in WMN. In [10] they investigate the throughput capacity of a WiMAX mesh tree, and they try to balance the impact of the depth of the tree with its fan-out.The approaches for node placements in WMN are depict in figure 1.
In this paper, we want to optimize WiMAX topology such that it meets some QoS constraints, especially throughput and delay, so we assume that all nodes in the network are in the transmission range and we do not consider the coverage provisioning in our work.
Using Genetic Algorithm (GA) in the field of network planning is widely investigated. Generally, the application is to optimize some network performance like capacity, topology and routing. In most of the works, the topology is graph, which has not the limitation of trees. Tree topologies have limitations dealing with GA operators that we will address in this paper.
The representation that widely used for coding trees in GA is Prufer numbers or Prufer sequence [11]. This encoding only represents trees, and each Prufer number represents exactly one tree. These interesting characteristics make it a good option, but it also has an important problem, low locality, which reduce the

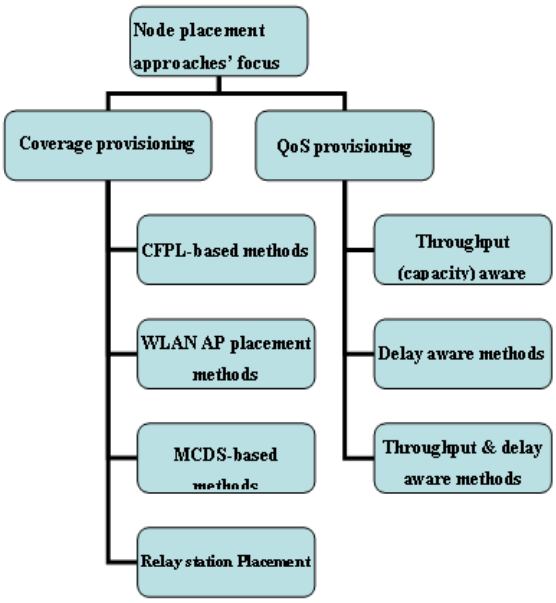


FIGURE 1: Node placement approaches

GA performance dramatically. In [12] the authors investigate the properties of Prufer numbers and show that it is a poor representation for trees.

3. ANALYTICAL MODEL

The network that we consider consists of N Mesh Subscriber Station (MSS) nodes and one Base Station (BS). Packets are forwarded in multihop fashion to or from the BS. We consider unidirectional traffic, i.e., the traffic that only goes from the MSS to the BS.

Each MSS_i has two queues, Q_r for relay packets and Q_s for packets originating from the node itself. The forwarding rules at each node is described as follows: [9]

1- If Q_r is empty, it sends one packet from Q_s.(Q_s always has packets to be transmitting)
2- If Q_r is not empty, it sends a packet from Q_r with a probability of q_i or a packet from Q_s with a probability of $1-q_i$.

q_i is forwarding probability for node i.

3.1 Throughput and Delay Analysis

We define per node throughput as the number of packets originating from node i successfully received by the BS. It can be obtained by counting the packets, which are received successfully by the BS without being blocked in any intermediate nodes. Thus, we need to calculate the blocking probability at each intermediate MSS. Based on queuing theory analysis, it is given as:

$$P_i^b = \begin{cases} \dfrac{[1-\rho_i]\rho_i^K}{1-\rho_i^{K+1}} & ,if\ \rho_i \neq 1 \\ \dfrac{1}{K+1} & ,if\ \rho_i = 1 \end{cases} \qquad (1)$$

Where ρ_i is the traffic intensity given above. Thus, we can state the throughput of node i, as below:

$$T_i = \begin{cases} \sigma_i^s & ,if\ |n_i^u| = 0 \\ \sigma_i^s . \prod_{i \in n_i^u} [1-P_i^b] & ,if\ |n_i^u| > 0 \end{cases} \qquad (2)$$

Where $|n_i^U|$ is number of nodes in the uplink path from node i to the BS. The average network throughput is obtained as follows:

$$mean(T) = \frac{\sum_{i=1}^{N} T_i}{N} \qquad (3)$$

To drive the delay which a packet from node i encounter, we need to compute the waiting time of a packet in each Q_r. For this we first obtain the steady state queue size of Q_r in node i as follows (using M/M/1/K analysis):

$$L_i^r = \begin{cases} \dfrac{\rho_i}{1-\rho_i} - \dfrac{\rho_i[K\rho_i^K+1]}{1-\rho^{K+1}} & ,if\ \rho_i \neq 1 \\ \dfrac{K(K-1)}{2(K+1)} & ,if\ \rho_i = 1 \end{cases} \qquad (4)$$

Then we will have the following expression for the waiting time of a packet in Q_r of node i:

$$W_i^r = \frac{1}{\mu_i^r} + \frac{L_i^r}{\lambda_r[1 - P_b^r]} \qquad (5)$$

Now, we can obtain D(i) by summing the waiting times spent in the intermediate nodes and the transmission times(i.e. τ), for traversing the i hopes:

$$D_i = \begin{cases} \tau & ,if\ |n_i^u| = 0 \\ \tau n_i^u . \sum_{i \in n_i^u} W_i^r & ,if\ |n_i^u| > 0 \end{cases} \qquad (6)$$

In addition, the average delay of the network can be found by taking into account the delay of packets, which have been successfully delivered:

$$mean(D) = \frac{\sum_{i=1}^{N} T_i D_i}{\sum_{i=1}^{N} T_i} \qquad (7)$$

4. PROPOSED GENETIC ALGORITHM

Given the number of nodes which exist in a WiMAX network; we intend to obtain a tree topology that satisfies some QoS metrics. For this purpose, we use a Genetic Algorithm (GA) approach. GA is one the most powerful approaches for optimizing complicated, multi-objective and large scale problems. Our problem in this paper is also, large scale and constrained. By a slight increasing in the number of nodes the search space grows dramatically and become too complicated to be addressed by conventional problem solving strategies, e.g. Linear Programming.
The proposed Genetic Algorithm is as followed:

Algorithm 1. Proposed GA

1. *Create a specified number of random keys with length l=n(n-1)/2.*
2. *For each individual produced in step 1, generate it's permutation sequence and it's corresponding spanning tree.*
3. *i=1*
4. *Do{*
    a. *At iteration i, evaluate fitness of each individuals.*
    b. *Select top 10 percent individuals from P_i and transfer them directly to population P_{i+1} , apply selection mechanism as defined in 4-2 to choose individuals.*
    c. *Apply cross over to generate offsprings.*
    d. *Apply mutation operator.*
    e. *Generate the next population P_{i+1}.*
    f. *Increment i.*
5. *} While a solution is not found or total number of generation is not reached or the function tolerance is less than threshold.*

Details of our algorithm are as follows:

4.1 Network Representation

We use NetKey representation which is simple and convenient to implement and characteristics of this representation is studied in [13].
The NetKey encoding belongs to the class of weighted encodings. In contrast to other representations such as CV encoding (Davis et al., 1993) which can only indicate whether a link is established or not, these encodings use weights for the genotypes and can thus encode the

importance of the links Also, an additional construction algorithm is necessary which constructs a valid tree from the genotypic weights.
For coding a tree, we first generate a random key sequence with length l=n(n-1)/2. Each element of this sequence which is between zero and one, describes the importance of the corresponding link (link ordering is like CV encoding).
The permutation r^s corresponding to a key sequence r of length l is the permutation δ such that δ_r is decreasing. These definitions say that the positions of the keys in the key sequence r, are ordered according to the values of the keys in descending order.
After generating a random key sequence and a permutation, we should construct a valid tree from that permutation. When constructing the tree, the positions of the key sequence r, are interpreted in the same way as for the CV.
The positions are labeled and each position represents one possible link in the tree. From a Key sequence r of length l, we have a permutation r^s ,of l numbers. Then the tree is constructed from the r^s as follows:

Algorithm 2. Tree Construction

1- *Let i=0 , T be an empty tree with n nodes, and r^s the permutation of length l=n(n-1)/2 that can be constructed from the key sequence r. All possible links of T are numbered from 1 to l.*
2- *Let j be the number at the ith position of the r^s.*
3- *If the insertion of the link with number j in T would not create a cycle, then insert the link in the T.*
4- *Stop, if there are n-1 links in T.*
5- *Increment I and go to step 2.*

The construction rule is based on Kruskal's algorithm (Kruskal 1956) and only consider the weights of the Random Keys vector for building the tree. With this rule, we can construct a unique, valid tree from every possible Random key sequence.

4.2 Selection and Reproduction

Based on trial and error we employ elitism to directly transfer top 10 percent individuals from current population to the next one. The rest of the population is created by crossing and mutating the genes. We use tournament selection with tournament size of four, for selecting the parents for mating pool. The reason why we employ this selection mechanism is to keep a balance between randomness and proportionality in search process.

4.3 Cross Over and Mutation Operator

Since the NetKey encoding have high locality, the standard cross over, mutation operators will work properly, and there is no need for problem specific operators. Because the operators will change the Random key sequence there will be no infeasible solution.

4.4 Fitness Function

With fitness function, we can evaluate the goodness degree of each individual in the population. For any desired QoS, we can have different fitness functions.
A well-known problem in wireless mesh networks is the fairness problem [5-7], i.e., the nodes farther away from the gateway may have a lower throughput than the nodes closer to the gateway. We address this problem here by searching for trees, which have smoothest throughput among its nodes. So in our fist scenario we want to find a tree t such that function f(t)=std(T) is minimized.
In the second scenario, we study the delay-minimization problem based on the analytical model described in 3. Here we want to find a QoS tree, which minimize the average delay in whole network. The fitness function here is f(t)=min(D).

5 NUMERICAL STUDY AND RESULTS

The network we choose for our numerical study is consisting of 18 nodes and one gateway. Based on Cayley(1889), there is exactly n^{n-2} possible tree for a graph with n nodes. This huge search space dare us to use GA, otherwise it will be too complex.

For simplicity we do not consider spatial reuse in the simulation, thus only one node is allowed to send within one time slot. The channel capacity is 75Mbs and the packet size is 1500B, so the length of time slot is set to 1500B/75Mbs=0.16 ms. The buffer size of Q_r is set to 30 for each node. In addition, for each node the channel access probability is assumed to be proportional to its sub tree and, probability for relaying packets is q=0.7 for all nodes. Population size is 50; crossover and mutation rates are adjusted to 0.8 and 0.02 respectively. Total number of GA iterations is set to 200.

The resulted tree topologies for fairness throughput and minimum delay are show in figure 2, 3 and the corresponding per-node throughput and delay for each node is depicted in figure 4(a), (b) respectively. As we can see, in the first scenario, the throughput among nodes is smooth and close to each other as we expected and the resulted tree is also almost balance. We know that for having a fair throughput among nodes especially for those that are far away from gateway we should consider tuning the forwarding probability for nodes,in the leaf nodes q=0 because there is no data to be forwarded but in intermediate nodes the q should be set to a value such that the node forward the childs traffic and also it's packets.

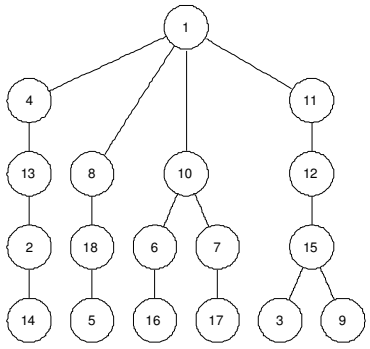


FIGURE 2: Resulted tree topology for fair Throughput

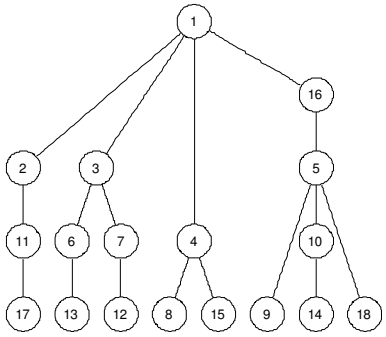
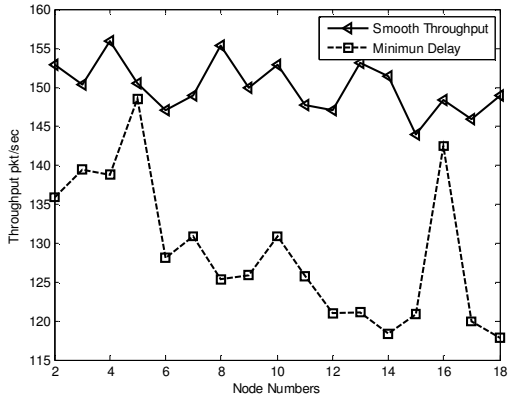


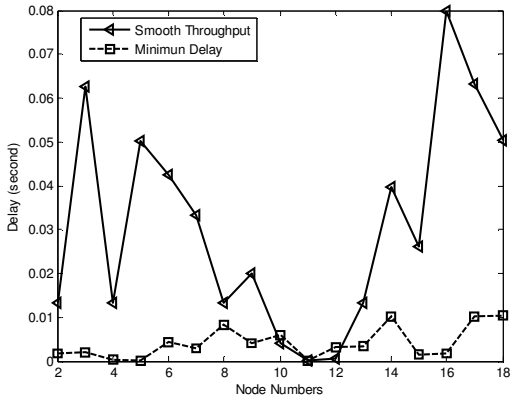
FIGURE 3: Resulted tree topology for minimum delay

In the second scenario the delay is minimized in the cost of decrease in throughput. As we can see in the resulted tree the depth is decreased and the node fan-out increased. Tuning forwarding probability (q) in this scenario is also important, if we set the q so high, the

intermediate nodes will always be busy by forwarding the leaf nodes data and their delay may increase or even they may starve.
We also change our scenario a bit to compare our proposed algorithm with [14]. We increase the number of nodes from 5 to 120 with step of 5 like [14] and change our fitness function to gain maximum throughput for each network.
For each network with n nodes we set the population size to $n^{1.5}$ based on the [13] and increase the number of generations based on the size of the problem from 100 to 700.
Figure 5 show the results from [14] and figure 6 show our result. We can see that our algorithm can find a topology with better overall throughput for networks from 5 to 65 nodes and the remain is almost the same.



(a)



(b)

FIGURE 4: Per-node delay and throughput of output tree, (a) per-node throughput, (b) per-node delay

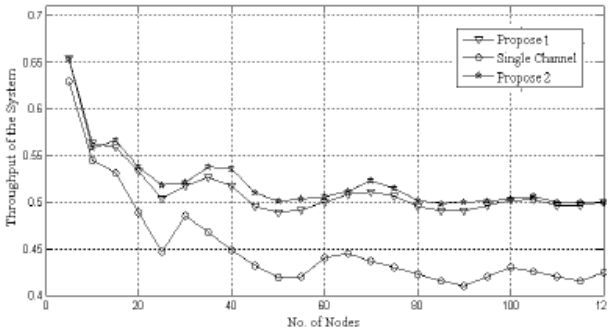


FIGURE 5: Resulted throughput from [14]

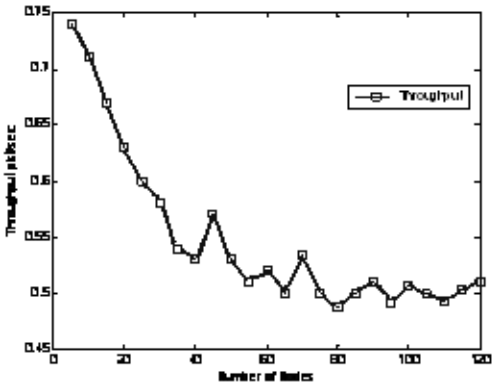


FIGURE 6: Maximum Throughput for different networks

6 CONCLUSION

In this paper, we propose a GA algorithm for finding a QoS tree for WiMAX mesh networks. Buffer size for each MSS node, transmission opportunity, lost packets, traffic load per node and delay in each node is considered. Our proposed GA algorithm is able to find QoS tree topology for given network. Any change in network parameter or nodes or application of network will not affect the algorithm at all. With defining proper fitness function this approach can always find the QoS tree with any desired delay and throughput. The obtained results show that different delay and throughput will lead to different tree's depth and fan out. For future works, node movements can be considered to overcome the limitation of the proposed algorithm, which are for the fixed nodes.

7 REFRENCES

1. IEEE 802.16d-2004,"Draft IEEE Standard for Local and Metropolitan area networks", May 2004.

2. S. Skiena, "Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica",Addison-Wesley,1990.

3. M. J. Lee, J. Zheng, Y.-B. Ko, and D. M. Shrestha, "Emerging standards for wireless mesh technology," *Wireless Commun",* 13(2): 56–63, 2006.

4. Salim Nahle, Luigi Iannone, Benoit Donnet and Naceur Malouch, "On the Construction of a wimax mesh tree", IEEE COMMUNICATIONS LETTERS, 11(12): 967 – 969, 2007

5. I. F. Akyildiz, X. Wang, and W. Wang, "Wireless mesh networks: A survey," *Comput. Netw.*, 47(4):445–487, 2005.

6. V. Gambiroza, B. Sadeghi, and E. W. Knightly, "End-to-end performance and fairness in multihop wireless backhaul networks," in *Proc. ACM MOBICOM*, Sep. 2004.

7. J. Jun and M. L. Sichitiu, "Fairness and QoS in multihop wireless networks," in *Proc. IEEE VTC*, Oct. 2003.

8. J.-F. Lee, W. Liao, and M.-C. Chen, "An incentive-based fairness mechanism for multi-hop wireless backhaul networks with selfish nodes," *IEEE Trans. Wireless Commun.*, 7(2): 697 - 704,2008.

9. T.Liu, W.Lio , "Location Dependent Throughput and Delay in Wireless Mesh Networks" , IEEE Transactions on Vehicular Technology, 57(2):1188-1198, 2008.

10. S.Nahle, L.Iannone, B.Donnet, and T. Friedman, “Investigating depth fanout trade-off in WiMAX mesh networks”, in Proc. 1st WEIRD workshop, May 2007.

11. H. Prufer. “Neuer Beweis eines Satzes uber Permutationen” Arch.Math. Phys.27(1) : 742-744,1918.

12. J. Gottlieb, G. R. Raidl, B.A. Julstrom, F. Rothlauf ,” Pr¨ufer Numbers: A Poor Representation of Spanning Trees for Evolutionary Search” IEEE Transactions on Evolutionary Computation, 4(2):125–134, 2002.

13. F.Rothluf , D.E. Goldberg and A.Heinzel, “Network Random Keys—A Tree Representation Scheme for Genetic and Evolutionary Algorithms” MIT press journals , 10(1): 75-97,2002.

14. Ali Al-Hemyari, Nor Kamariah Noordin, Alyani Ismail, Sabira Khatun, Yaseen H. Tahir, and Y.A. Qassem “Centralized Scheduling, Routing Tree in WiMAX Mesh Networks”, Innovations in Information Technology, 2008. IIT 2008, Al Ain UAE.

An Algorithm for Computing Average Packet Delay

M. R. Hassan m_r_hassan73@yahoo.com
Computer Science Department,
Faculty of Science, South Valley University,
Aswan, Egypt.

Abstract

Average Packet delay is considered as a vital performance measure for a computer-communication network especially in the network designing problem. Average Packet delay evaluation depends on two main parts: the first part is the capacity of each link in the network, the last one is the flow of each link. The capacity of each link is assumed to be fixed but the flow of each link is computed by using routing algorithms and the traffic requirement matrix. The paper presents an algorithm based on FLOYD's routing algorithm to calculate the flow of each link and then we can compute the average packet delay.

Keywords: Computer Networks, QoS, Average Packet Delay.

1. INTRODUCTION

Performance measures [1], eg, delay, throughput are very important measures in designing reliable networks. The performance measure, Average Network Throughput (ANT), which takes into consideration the network topology, link reliabilities, the capacity of the links in the network, and the total installed capacity, [2] and [3]. The optimization priblem of ANT has been studied by many researches, [4-6].

The other vital performance measure for a communication computer network is the average packet delay, T, that is, the mean time that a packet takes to travel from a source node to a destination node in the network, [7]. The average packet delay optimization problem formulated and solved in [8-10]. The average source-to-destination packet delay is considered as an important criteria in the computer network design problems [11].

This paper presents a simple algorithm to calculate the average packet delay (T) of a given computer network by generating the set of all shortest paths by using Floyd's algorithm, [12], to determine the route to be taken by packets between each source-destination pair of nodes based on Euclidean distance between them. Then assumed the traffic that has been actually carried by each link in the network to calculate the flow of this link.

The paper is organized as follows: The assumptions and notation used given in Section 2. Section 3 describes the problem of calculating the average packet delay. Section 4 presents the proposed algorithm for calculating the average packet delay. Section 5 shows how to use the proposed algorithm to calculate the average packet delay for a given network. Conclusion and future work are given in section 6.

2. NOTATIONS AND ASSUMPTIONS

Notations:

N is the number of nodes.
M is the number of links in the network.
C_k is the capacity of link k.
F_k is the flow of link k.
L_k is the length of link k.
γ_{ij} is the traffic between nodes i and j.
γ is the total traffic in the network.
$1/\mu$ is the average packet length.

Assumptions:

1. The location of each network node is given.
2. The traffic between each node-pair has a Poisson distribution.
3. The packet size has exponential distribution with mean $1/\mu$ (bits/packet).
4. The nodal memory is infinite.
5. Independence between interarrival and transmission times on each link.

3. PROBLEM DESCRIPTION

The computer network is modeled as a directed graph G (N, L), |N| = n and |L| = *m*, where n represents the number of nodes of the network and *m* represents the number of links of the network. If we consider the sample network of 4 nodes and 5 links (n = 4 and *m* = 5) as shown in Fig. 1.

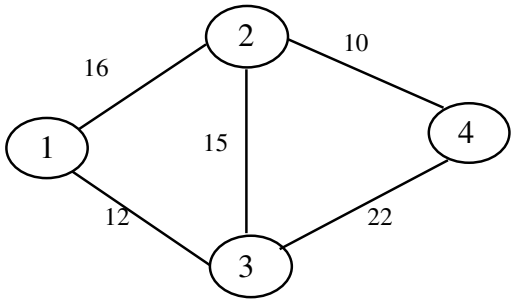


FIGURE 1: A sample network

The distance matrix is:

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 16 & 12 & 0 \\ 16 & 0 & 15 & 10 \\ 12 & 15 & 0 & 22 \\ 0 & 10 & 22 & 0 \end{bmatrix} \end{array}$$

FIGURE 2: The Distance matrix for the network in Figure1.

The shrtest paths are as follows by using Floyd's algorithm, [12] :-

P_{12} = 1-2
P_{13} = 1-3
P_{14} = 1-3-4
P_{23} = 2-3
P_{34} = 3-4
P_{24} = 2-4

The flow of each link can be computed by summed the packets that travel across that link. The total number of packets that carried by the link depends on the number of shortest paths contains that link. The following table summerizes the link and the paths that contains that link.

The link	The paths
1-2	Appears in P_{12}
1-3	Appears in P_{13} and P_{14}
2-3	Appears in P_{23}
2-4	Appears in P_{24}
3-4	Appears in P_{13} and P_{34}

TABLE 1: The links and the corresponding paths

So, if the number of packets equal *p* then the following table summerize the number of packets carried by each link.

The link	The number of packets
1-2	*p*
1-3	*2p*
2-3	*p*
2-4	*p*
3-4	*2p*

TABLE 2: The number of packets carried by each link

So, the total flow is equal to *14p*, if we consider the link is bidirectional.

In the following subsections we will describe the main parts of the algorithm to calculate the flow of a given network and then calculate the average delay.

3. 1. Shortest Path Generation

We wil use Floyd's all-pairs shortest path algorithm, [12], to find all shortest paths from the source node to the destination node by using the distance matrix as follows:

SubAlgorithm_1

Input:
V = set of vertices, labeled by integers 1 to N.
E = set of edges, labeled by ordered pairs (*u, v*) of vertex labels.
D[u][v]: The distance matrix. *D[u][v]=0* if $u \neq v$ and $(u, v) \notin E$ or *u=v*
P[u][v]: the shortest path from *u to v* . *P[u][v]=u and P[u][v]=0* if $u \neq v$ and $(u, v) \notin E$ or *u=v*.

For all edges (*u, v*) in E:
W[u][v] = D[u][v].. And *W*(*u, v*) = infinity if $u \neq v$ and $(u, v) \notin E$ or *u=v*.

```
Begin
 for(k =1; k<= n; k++)
  for(u =1; u<= n; u++)
    for(v =1; v<= n; v++)
        W[u][v] = min (W[u][v], W[u][k]+ W[k][v])
          P[u][v]=k;
end.
```

3. 2. Finding the Intermediate Nodes for Each Shortest Path

If $P(i, j)=V_q$ then intermediate nodes on shortest path from i to j can be deduced as follows:

SubAlgorithm_2:

```
for(i=1;i<n;i++)
  for(j=i+1;j<=n;j++)
          Begin
           if(p[i][j]!=i) then
          repeat
                    q++;
                     v[q]=p[i][v[q-1]];
          until v[q]=v[q-1]
    for(k=1;k<q;k++) print  v[k];
End
```

3. 3. Calculate the Flow of Each Link

For each path generated in 3.2 count the total traffic that carried by each link connects the corresponding pair of nodes:

SubAlgorithm_3
Initially set flow[][]=0;
For each path generated in 3.2 do
Begin
If the path contains q vetices then
For(k=1; k<q,k++)
flow[i][V[k]]+= $\gamma_{iv[k]}$;
END

3. 4. The Total Traffic
The total traffic across the network is calculated by:

$$\gamma = \sum_{i,j}^{n} \gamma_{ij} \qquad \ldots(1)$$

SubAlgorithm_4:
Set γ = 0
for(i=1;i<n;i++)
for(j=i+1;j<=n;j++)
$\gamma = \gamma + \gamma_{ij}$

3. 5. The Average Packet Delay
The Average packet Delay, the mean time that a packet takes to travel from a source node to a destination node in the network is given by [11]:

$$T = \frac{1}{\gamma} \sum_{i=1}^{m} \frac{f_i}{c_i - f_i} \qquad \ldots.(2)$$

4. AN ALGORITHM FOR COMPUTING THE AVERAGE PACKET DELAY

The steps of the algorithm for calculating the average delay of the computer network are as follows:

Step 1: Read the distance matrix D and the capacity for each link C_i
Step 2: Generate shortest paths using Sub*Algorithm_1* and deduce the intermediate nodes for each path using SubAlgorithm_2.
Step 3: Use SubAlgorithm_3 to calculate the flow of each link, f_i
Step 4: Calculate the total traffic, γ by using Sub*Algorithm_4.*
Step 5: Calculate the Average Packet Delay, T, using equation (2).

Note: The algorithm has been implemented using VC++ 6.0.

5. CASE STUDY

To illustrate the proposed algorithm for computing the Average Packet Delay, consider an example networks taken from [9]. As shown in figure 2 the network has 8 nodes 13 links. The numbers written in bold represent the link lengths.

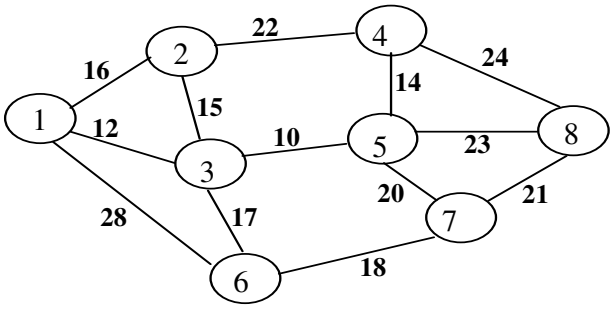


FIGURE 3: Example network

Table 3 shows the link flows (computed by the proposed algorithm) and corresponding capacities (taken from [10]) for the given network shown in Figure 1.

Link	Flow (Kbps)	Capacity (Kbps)
1 - 2	10	19.2
1 - 3	50	56
1 - 6	10	19.2
2 - 3	40	56
2 - 4	20	56
3 - 5	100	200
3 - 6	40	100
4 - 5	30	56
4 - 8	20	56
5 - 7	50	56
5 - 8	30	56
6 - 7	20	56
7- 8	20	56

TABLE 3: Link flows and Capacities

The total traffic, $\gamma = 65.$

The Average Packet delay (T) is equal to 0.419 s

The following table shows the link flows (computed by the proposed algorithm) and corresponding capacities for the given network with 10 nodes and 14 links shown in Figure 2, taken from [8].

Link	Flow (Kbps)	Capacity (Kbps)
1 - 2	80	100
1 - 5	220	230.4
2 - 3	100	230.4
2 - 4	60	100
3 - 8	140	230.4
4 - 5	60	100
4 - 8	80	100
5 - 6	100	230.4
5 - 7	180	230.4
6 - 7	60	100
6 - 10	80	100
7 - 9	140	230.4
8 - 9	160	230.4
9 - 10	100	230.4

TABLE 4: Link flows and Capacities

The total traffic, $\gamma = 140$.
The Average Packet delay (T) is equal to 0.3423 s

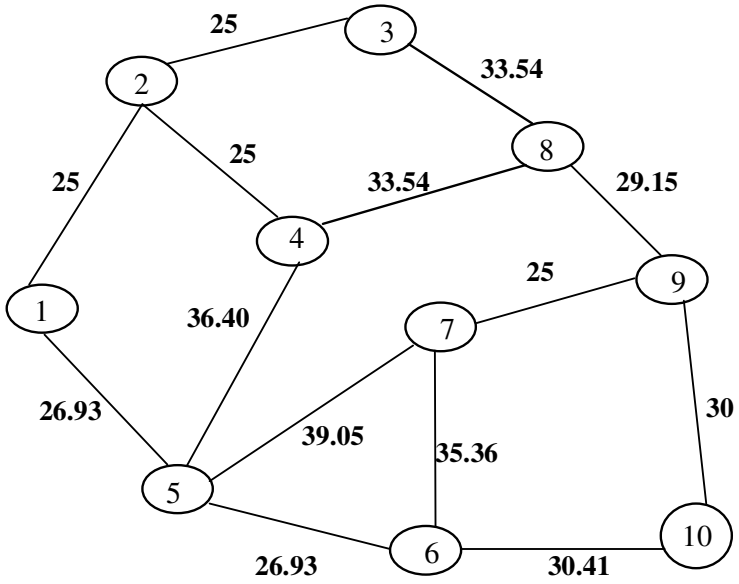


FIGURE 4: Example network taken from [8].

The above results shows that the proposed algorithm is working properly and efficient for the computer networks with large number of nodes.

6. CONCLUSION AND FUTURE WORK

This paper presented a simple algorithm to calculate the average packet delay. The algorithm is based on determining the flow of each each link of a given network. The flow claculations depends on the shortest path generated by the routing algorithm. Finally we illustrate the using of the proposed algorithm by calculating the average packet delay to a given sample network. In the future work we hope that the algorithm can be used in the average delay optimization problems.

7. REFERENCES

1. Tokumi Yokohira, Masashi Sugano and Hideo Miyahara, “Fault Tolerant Packet-Switched Network Design and Its Sensitivity”, IEEE Transactions on Reliability, 40(4):452-460, 1991.

2. Ahuja Sanjay P. and Kumar Anup “Reliability and Performance based measure for computer networks and distributed systems “, IEEE Southeastcon, Charlotte, NC,1993.

3. M. R. Girgis, A. Younes and M. R. Hassan, “An Algorithm for Computing Throughput of Computer Networks”, Egyptian Informatics Journal, 9(1):205-218, 2009.

4. Ahuja Sanjay P., “Performance based reliability Optimization for computer networks “, Engineering the New Century Conference Proceedings-IEEE-Southeastcon. IEEE, Piscataway, NJ, USA, 97CB36044, PP. 121-125, 1997.

5. Kumar Anup, Elmaghraby Adel S. and Auja Sanjay P., "Performance and reliability optimization for distributed computing systems", 3rd IEEE Symposium on Computers and Communications, Athens, Greece, pp. 611-615, 1998.

6. M. R. Girgis, A. Younes and M. R. Hassan, “Optimizing The Performance-Based Reliability For Computer Networks By Using Fuzzy Optimization With Genetic

Algorithms", International Journal of Computational and Applied Mathematics, 2(2):139–148, 2007.

7. S. Pierre and A. Elgibaoui," Improving Communication Network Topologies Using Tabu Search", Proceedings of 22nd Annual Conference on Local Computer Networks, pp. 44-53, 2-5 Nov 1997.

8. R. Beaubrun and S. Pierre, " A Routing Algorithm for Distributed Communication Networks", Proceedings of 22nd Annual Conference on Local Computer Networks, pp. 99-105, 2-5 Nov1997.

9. Peirre Samuel and Legault, "A genetic algorithm for designing distributed computer network topologies", IEEE Transactions Systems , MAN and Cybernetics-Part B: Cybernetics, 28(2):249-258, 1998.

10. M. R. Girgis, A. Younes and M. R. Hassan, "Optimizing the transmission delay of a computer network by using fuzzy optimization with genetic algoritms", International Journal of Intelligent Computing and Information Science, 8(1):163-171, 2008.

11. Gerla, M., and Kleinrock. L. "On theTopological Design of Distributed Computer Networks". IEEE Transactions on communications,25(1): 48-60,1977.

12. Maclej M. Syslo, "Discrete Optimization Algorithms: With Pascal Programs", Prentice Hall, (1983).

RRSTP: A Spanning Tree Protocol for Obviating Count-to-Infinity from Switched Ethernet Networks

Syed Muhammad Atif syed.muhammad.atif@gmail.com
M.S Computer Networks
Department of Computer System Engineering
Usman Institute of Technology
Karachi, Pakistan.

Abstract

This paper proposes a highly reliable and rapidly converging spanning tree protocol named as Reliable Rapid Spanning Tree Protocol. The need of this spanning tree protocol is felt because reliability of switched Ethernet networks is heavily dependent upon that of spanning tree protocol. But current standard spanning tree protocol – Rapid Spanning Tree Protocol – is well known for its susceptibility to classical count-to-infinity problem. Because of this problem the protocol has extremely variable and unexpectedly high convergence time even in small networks. As a result network wide congestion, frame loss and frame delay may occur. Even forwarding loops may be induced into the network under certain circumstances. It is expected that the new protocol – RRSTP – will significantly increase the dependability of switched Ethernet networks by providing guaranteed protection against the count-to-infinity problem.

Keywords: Network Reliability, Count-to-Infinity, Network Convergence, RSTP.

1. INTRODUCTION

For last two decades, switched Ethernet networks are the most popular local area networks (LANs). The reasons of it are its auto configuration, easy availability, low cost, backward compatibility and scalability to higher bandwidth. In switched Ethernet networks, switches are core devices. Ethernet switch is a multi-port layer 2 network device that forwards frame to specific ports rather than, as in conventional hub, broadcasting every frame to every port. In this way, the connections between ports deliver the full bandwidth available. That is why switched Ethernet networks exhibit appreciably better performance, throughput and scalability than that of conventional Ethernet networks.

In its pure form switched Ethernet networks cannot be used with a physical topology having cycles or redundant links. The reason is two folded. First, broadcast frames and unknown unicast frames flooded by switches may circulate in cycle forever. Second, dynamic address learning mechanism may pollute the filtering table of the switch. Since redundant links in physical topology are highly essential for fault tolerance that is why most present switches use a vital management protocol known as spanning tree protocol in order to allow physical topologies having redundant link or cycles. This protocol puts redundant links of physical topology in hot standby position by developing a logical spanning tree, in distributed fashion, over an underlying physical topology. Thus the physical topology seems to be cycle free for all switches in the network. Current standard spanning tree protocol, commonly known as Rapid Spanning Tree Protocol, is a variant of distance vector routing protocol. Distance vector routing protocols and so RSTP [1] are consider highly vulnerable to count-to-infinity problem. But it was Mayer at al. [2] who first mentioned that count-to-infinity problem may become severe under certain circumstances in RSTP controlled network. This highly undesirable behavior were later studied in detail by Elmeleegy at el. [3] [4] and Atif [5]. When count-to-infinity occurs, convergence time of RSTP sharply increased to tens of seconds [3]. Moreover, forwarding loops may also be induced into the network to further complicate the problem [4]. This vulnerability of RSTP severely affects the

reliability of switched Ethernet networks as their reliability is heavily dependent on their spanning tree protocols. This paper will present an all new spanning tree protocol – named as Reliable Rapid Spanning Tree Protocol – specifically tailored to provide protection to Ethernet network against highly undesirable count-to-infinity problem. RRSTP is designed in such a manner that its convergence time is comparable to that of RSTP. Therefore, the new protocol will dramatically increase the reliability of switched Ethernet networks without compromising on their availability.

The rest of paper is organized as follows. Section 2 will give a brief overview of RSTP [1]. Section 3 will discuss the conditions that need to be satisfied for count-to-infinity to occur in a spanning tree protocol controlled network. Section 4 will propose the solution to handle this problem. Section 5 will discuss the related work and then section 6 will conclude the paper.

2. OVERVIEW OF RAPID SPANNING TREE PROTOCOLS

Current standard spanning tree protocol – Rapid Spanning Tree Protocol – is the successor of Spanning Tree Protocol. The earlier standard Ethernet spanning tree protocol – STP – was first proposed by Perlman in [6]. RSTP [1] is basically an integration of work of Mick Seaman presented in [7], [8], [9], [10] to reduce the convergence time of STP [11]. This section gives a brief overview of RSTP.

In RSTP every switch and every port of a switch has a unique identifier. A Root Switch, a switch having the smallest switch identifier, is elected through a distributed mechanism. Each switch calculates and maintains the shortest path to the Root Switch to construct the spanning tree. Switch and Port Identifiers are used as tie breaker when two paths are otherwise same.
Switches use Bridge Protocol Data Units (BPDUs) to exchange information among them. A port that is receiving the BPDU having the best path to Root Switch becomes the root port of the switch. All remaining ports of a switch always transmit the BPDUs having information of switch's root port. Ports receiving inferior information than one they are transmitting become designated ports. A switch uses only its root port and designated ports for forwarding data. Alternate and back up ports, ports that are neither the root port nor designated ports, are kept in stand by position for use in case of link failure or topology change.

RSTP has several unique features to keep the convergence time as low as possible. In RSTP, an alternate port, a port that have a better but not the best path to the Root Switch, becomes the new root port and so immediately moves into forwarding state after retirement of the current root port [7]. For quick propagation of failure information, an RSTP switch is allowed to process inferior BPDUs on the root port and alternate ports, if they are transmitted by their respective designated port [8]. Further, RSTP uses a handshake mechanism (sync) to quickly put a designated port, connected to a point-to-point link, into forwarding state [9].

In the event of a topology change, switches need to flush some of their addresses from its forwarding table, a table that records MAC addresses and their associated ports learnt through address learning mechanism. It is necessary because a station may change its position with respect to switch after a topology change. RSTP uses an address flushing mechanism presented by Vipne Jain and Mick Seaman [10]. This mechanism can flushes the required addresses more quickly as compared to the one used by STP.

3. COUNT-TO-INFINITY IN SPANNING TREE CONTROLED NETWORKS

This section will discuss when and how count-to-infinity may occur in spanning tree controlled networks. Count-to-infinity problem is highly undesirable to Ethernet networks as it adversely effects the convergence time of the network and thus decreases the network availability. Atif, in [5], has deeply discussed the count-to-infinity problem in spanning tree controlled networks in a novel fashion using some new terminologies. In that paper he has mentioned six conditions that must be satisfied simultaneously in order to induce count-to-infinity into the network. This section will partially reproduce the original work of Atif in [5] for ease of reference.

In a fully converged spanning tree controlled network all alternate ports are dual rooted i.e. have two distinct path to the Root Switch. One path of an alternate port to the Root Switch passes through its link's designated port while the other path passes through its switch's root port. However an alternate port may loss its one or both paths to the Root Switch if the root port of its upstream switch fails. So in a network in which a switch suffering from the root port failure, an alternate port may have no, one or two path(s) to the Root Switch and thus will be called orphan, single rooted and dual rooted alternate port respectively in this text. Orphan alternate ports must not be used to reunite the network temporarily segregated due to the root port failure of a switch. Because such alternate ports have information which is no longer valid. Moreover, dual rooted alternate ports are not used by spanning tree protocols to prevent forwarding loops. This left only single rooted alternate ports that can be used reunite the partitioned network, which in fact also have the potential to do so. Hence the underlying spanning tree protocol must use only single rooted alternate ports to restore connectivity.

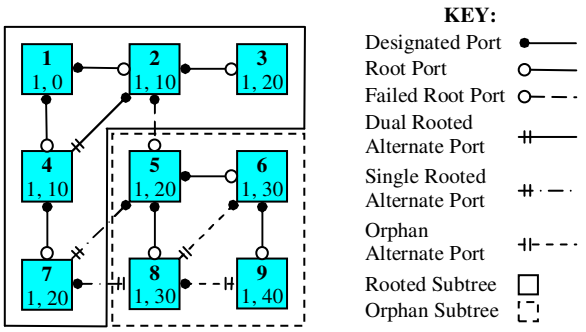


FIGURE 1: Different types of alternate ports in a network after failure of the root port of switch 5.

In a fully converged spanning tree controlled network, failure of the root port (or the designated port associated with the root port) of a switch results into a partition of underlying spanning tree into two distinct subtrees namely a *rooted subtree*, a subtree that still have the Root Switch, and an *orphan subtree*, a subtree that no longer have the previous Root Switch. It has to be noted that since all the switches in the orphan subtree have lost their path to previous Root Switch through their respective root ports. Therefore dual rooted alternate ports cannot exist in orphan subtree. In contrast, all the switches in rooted subtree have a path to the Root Switch through their respective root port. Hence orphan alternate ports cannot exist in rooted subtree. However, single rooted alternate ports can be found in both subtrees but only near the common boarder of these two subtrees. An alternate port in the rooted subtree is single rooted if and only if its associated designated port is in the orphan sub tree. Similarly an alternate port in the orphan subtree is single rooted if and only if its associated designated port is in the rooted subtree. These facts are depicted in Figure 1 through an exemplary network. Each switch is represented by a small box. The top number in the box is the Switch ID, the lower set of numbers represents the Root Switch ID as perceived by the switch and the cost to this Root Switch. All links have cost of 10. Figure 1 shows the snapshot of network immediately after failure of the root port of switch **5**. Switches **1** to **4** and switch **7** are in rooted subtree and switch **5**, **6**, **8** and **9** are in orphan subtree. Alternate port of switch **4** is still dual rooted as it is inside the rooted subtree. Moreover, Alternate port of switch **7** and that of switch **8** connected to switch **7** are single rooted alternate ports as they are near the common boarder of two subtrees. While alternate of switch **8** connected to switch **6** and that of switch **9** are orphan alternate ports as they are inside the orphan subtree.

Switches in a spanning tree controlled network use messages to communicate with each other. These messages experience a transmission delay when passing through the network. Thus, failure the root port of a switch may put all its downstream switches, that is switches in orphan subtree, into an inconsistent state for a period of time. The absolute period of inconsistence for a

switch **B** is from the time when one of its upstream switch's root port (or the designated port associated with the upstream switch's root port) fails to the time when this information will be received on the root port and all alternate ports (if any) of the switch **B**. The effective period of inconsistence for a switch **B** is a bit small and it spans from the time when the first time switch **B** receives failure information of its upstream switch's root port on its root (or alternate) port to the time this will be received on all its remaining alternate port(s) (and the root port). Clearly, only inconsistent switches may have orphan alternate port(s) because of lack of information. Further, such switches cannot differentiate an orphan alternate port from the other two types of alternate ports.

Count-to-infinity occurs in the part of network constituting the orphan subtree, if six conditions are satisfied simultaneously. Three of them have to be satisfied by an inconsistent switch **B**:

1. Switch **B** has an orphan alternate port **a** such that its root path cost is smaller than that of the best single rooted alternate port in the network.
2. Switch **B** starts to declare its orphan alternate port **a** as designated port or the root port when it is still in the effective inconsistent port or switch **B** is declaring its orphan alternate port **a** as designated port when it is entering into the absolute inconsistent state.
3. Switch **B** is injecting the stale BPDU through its retiring orphan alternate port **a** that is becoming designated port or through its retiring root port that is becoming the designated port because the orphan alternate **a** is becoming the new root port.

Two conditions must be satisfied by an upstream switch **A** along with above three conditions:

4. Switch **A** accepts the stale BPDU, transmitted by switch **B**, on its designated port **d**, as it is conceived as superior BPDU by switch **A**. This makes port **d** the new root port of switch **A**. It may happen only if the switch cannot differentiate between stale and fresh BPDUs.
5. Switch **A** begins to propagate the stale BPDU further through its now designated ports.

One condition needs to be met by underlying network.

6. There is at least one (unbroken) cycle in the network passing through switch **A**'s new root port **d** and switch **B**'s orphan alternate port a.

The first and the last condition for count-to-infinity are unavoidable in a high available fault tolerant network. However, remaining conditions can be easily avoided from being satisfied, by making slight modifications in underlying spanning tree protocol, to make the underlying network completely secure from the highly dangerous count-to-infinity problem.

When count-to-infinity occurs, the stale information begins to circulate in cycle and thus increments the root path cost of suffering switches with a definite offset, equal to the cycle's path cost, in each complete cycle (see Figure 2). Theoretically speaking, count-to-infinity in the network may be temporary or absolute. Temporary count-to-infinity in the network terminates after a definite interval of time. On the other hand absolute count-to-infinity persists forever. Temporary count-to-infinity occurs when the network is temporary segregated i.e. there is at least one single rooted alternate port in the network but a downstream switch mistakenly turns its orphan alternate port into root or designated port to reunite the partitioned network. When this happen count-to-infinity lasts until root path cost of one of the suffering switch exceed to that of the best single rooted alternate port in the network. Absolute count-to-infinity occurs when the network is absolutely segregated ,that is there is no single rooted alternate port in the network or in other words the best single rooted alternate port in the network has the root path cost of infinity, but a downstream switch starts to use an orphan alternate port to reunite the partitioned network.

Backup port can be made designated port after failure of its corresponding designated port without any fear of induction of count-to-infinity into the network. The reason is two folded. First, all the root ports on the shared medium start to pretend like single rooted alternate ports that can provide a path to Root Switch through the backup port corresponding to the failed designated port. Second, the root path cost of these pretending single rooted alternate ports is better than that of all orphan alternate ports in the orphan subtree i.e. violation of condition 1 of six conditions

required for count-to-infinity. Moreover, change in port cost of the root port of a switch also forces the port to act like a single rooted alternate port.

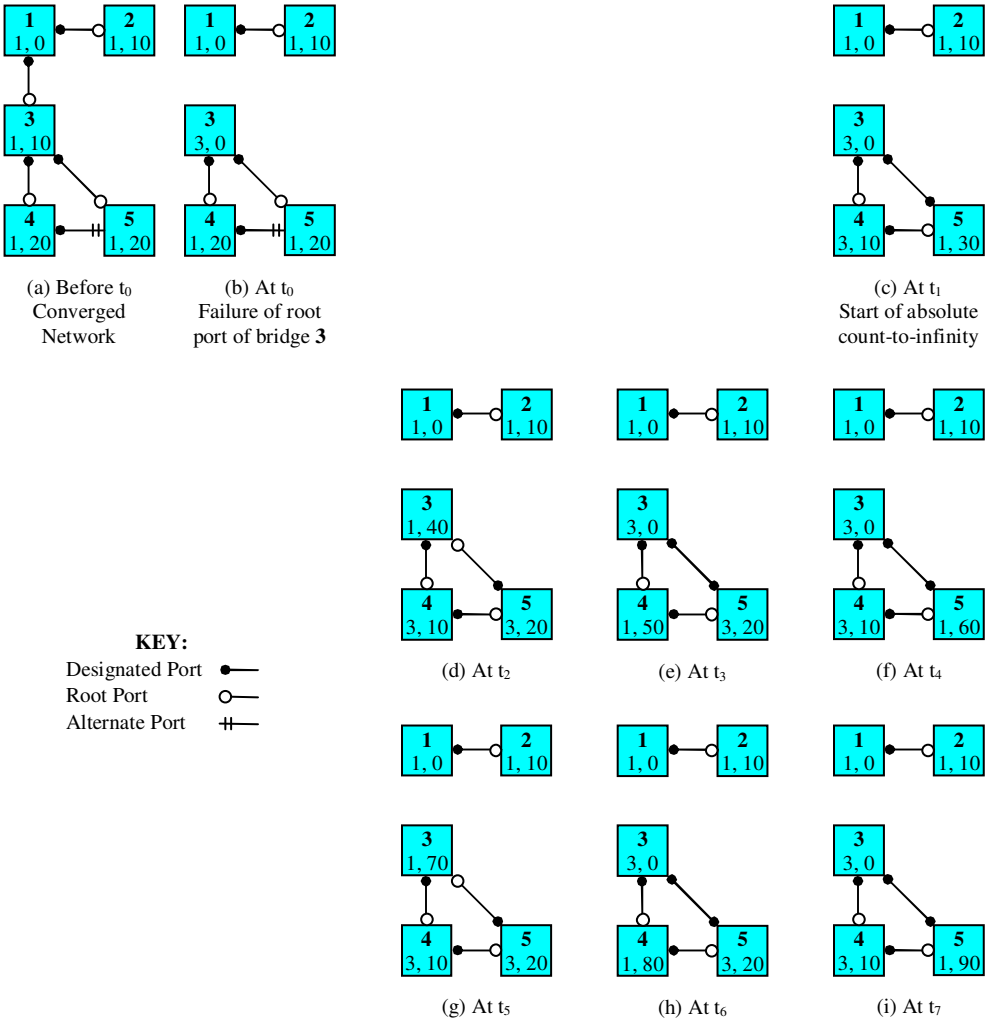


FIGURE 2: A network suffering from absolute count-to-infinity after failure of switch 3's root port because switch 5 is declaring its orphan alternate port as the new root port.

In RSTP controlled switched Ethernet networks above mentioned six conditions for count-to-infinity may be satisfied quite easily. That is why such networks are highly vulnerable to both temporary and absolute count-to-infinities. This highly undesirable behavior of RSTP was discussed, in detail, by Atif in [5].

4. RRSTP: THE RELIABLE RAPID SPANNING TREE PROTOCOL

Reliable Rapid Spanning Tree Protocol – RRSTP – is a spanning tree protocol that is specifically designed to protect switched Ethernet networks against count-to-infinity. The protocol consist of two mechanisms namely Rapid BPDU Distribution Mechanism and Root Switch Reelection Mechanism for timely convergence of network to a new topology after a topological change. This section will discuss the operation of RRSTP in detail.

Rapid BPDU Distribution Mechanism

It is mentioned in section 3 that switches in orphan subtree cannot distinguish between single rooted and orphan alternate ports when they are in effective inconsistent state. However, in RSTP, switches in orphan subtree are allowed to use their alternate ports to keep the convergence time as low as possible. This opens a gate for induction of count-to-infinity into the network. In contrast, RRSTP does not allow switches to use their alternate ports. But to keep the convergence time comparable to RSTP, an alternative mechanism is provided in RRSTP and it is named as Rapid BPDU Distribution Mechanism. It is the core or primary convergence mechanism used by RRSTP.

The Rapid BPDU Distribution Mechanism, in its simplified form, works as follows:

1. If a non edge designated port of a switch fails (that may be marked as failure of neighboring root port on the link) then
   a. Transmits Configuration BPDU on all its non-edge designated port(s).
   b. Transmits a Request BPDU on its root port.
2. If a switch receives an Request BPDU on its non-edge designated port then it must do a and b of 1.

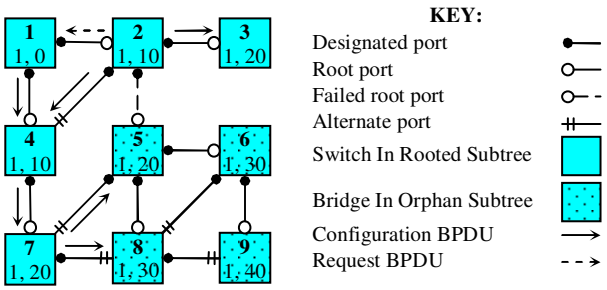


FIGURE 3: Flow of Configuration and Request BPDU after failure of the root port of switch 5 in an RRSTP controlled Ethernet network using Rapid BPDU Distribution Mechanism.

Upon failure of designated port of a switch, all the descendent (downstream) switches of that port become part of an orphan subtree. But the switch experiencing the designated port failure is still in the rooted subtree. That is why that switch initiates the Rapid BPDU distribution mechanism in RRSTP. Configuration BPDUs propagated through this mechanism may enter into an orphan subtree only through single rooted alternate ports as depicted in Figure 3. So, switches in an orphan subtree may use such BPDUs without any fear of induction of count-to-infinity.

Configuration BPDU in RRSTP, similar to RST BPDU in RSTP, has two new fields namely Sequence Number and Originator Root Path Cost fields to facilitate switches in an orphan subtree in recognizing fresh (valid) BPDUs transmitted through Rapid BPDU Distribution Mechanism. Only Root Switch is allowed to decrement Sequence Number. Non-root switches can generate BPDUs with the latest Sequence Number it has received from the Root Switch. Switches in orphan subtree must discard a BPDU if the BPDU has Sequence Number worse than they already have. Further, a switch in orphan subtree can accept a Configuration BPDU with same Sequence Number if and only if its Originator Root Path Cost is better than that conceived by receiving switch. A non-root switch can generate Configuration BPDUs only with Originator Root Path Cost worse than or equal to its own Root Path Cost. However, it is perfectly allowed for a switch to relay BPDUs with Originator Root Path Cost better than its own Root Path Cost.

Root Switch Reelection Mechanism

In RRSTP, switches in orphan subtree are solely dependent upon switches in the rooted subtree for converging to a new topology. However, Configuration BPDUs generated through Rapid BPDU Distribution Mechanism cannot enter into an orphan subtree if the network is absolutely segregated i.e. the network has no single rooted alternate port. This may force switches in an

orphan subtree to stick to previous topology indefinitely. To overwhelm this problem, RRSTP has Root Switch Reelection Mechanism. This mechanism is used only as backup or secondary mechanism for converging the network.

The Root Switch Reelection Mechanism, in its simplified form, works as follows

1. If the root port of a switch fails then
   a. Set the mode of the switch to Inconsistent
   b. Start Inconsistent Mode timer
2. If a switch receive a fresh (valid) BPDU with Consistent Flag is clear on its root port then do a and b of 1.
3. If a switch in Inconsistent mode receives a fresh (valid) BPDU with Consistent Flag is set then it reverts back to Consistent mode again.
4. If the Inconsistent Mode timer of switch expires and the switch is still in the Inconsistent Mode then the switch must declare itself the Root Switch and move into Consistent mode again.

RRSTP adds a new field that is Network Identifier into the Configuration BPDU. A non-root switch in an orphan subtree decrements its Network Identifier field before declaring itself the root switch after expiration of its Inconsistent Mode Timer. It makes fresh (valid) BPDUs, originated by that switch, more preferable over stale (invalid) BPDUs, previously originated by the now inaccessible Root Switch. An incoming BPDU is accepted if Network Identifier in the BPDU is less than that conceives by the receiving switch. It ensures that stale BPDUs of the now inaccessible Root Switch will not override fresh BPDUs announcing a switch of orphan subtree as the Root Switch.

Protocol Definition

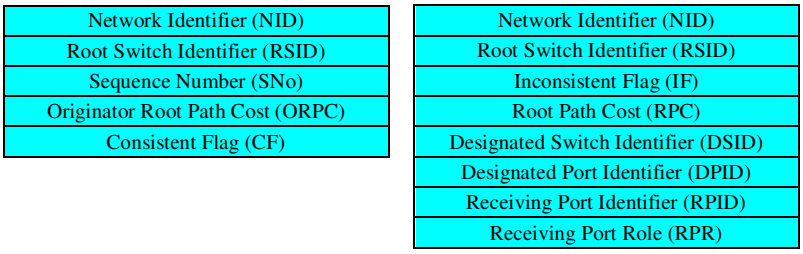
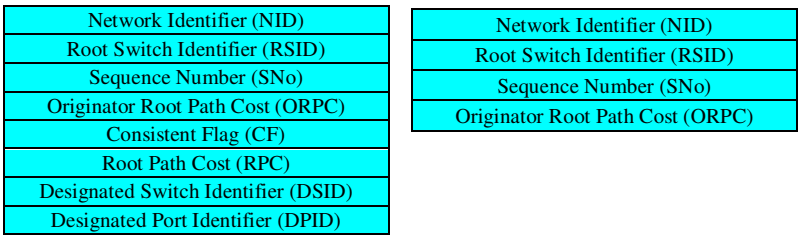


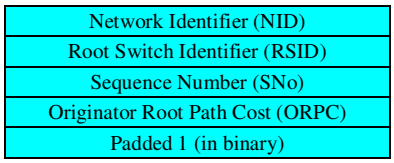
FIGURE 4: Structure of Priority Vector Pair in RRSTP.

Operation of RRSTP can be defined precisely and concisely with the help of six pairs of Priority Vectors namely Switch Priority Vector Pair, Root Priority Vector Pair, Root Path Priority Vector Pair, Port Priority Vector Pair, Designated Priority Vector Pair and Message Priority Vector Pair. Each Priority Vector Pair consist of a Network Priority Vector and a Configuration Priority Vector. Figure 4 is showing the structure of a Network Priority Vector and a Configuration Priority Vector in a Priority Vector Pair. Further, RRSTP uses two types of BPDUs that is Request BPDU and Configuration BPDU for communication with neighboring switches. Structure of Configuration and Request BPDUs are shown in Figure 5. A Request Priority Vector is also used in RRSTP to store and process Request BPDUs (See Figure 6 for structure of Request Priority Vector).



(a) Configuration BPDU (b) Request BPDU

FIGURE 5: Structure of Configuration and Request BPDUs in RRSTP.



(a) Request Priority Vector

FIGURE 6: Structure of Request Priority Vector in RRSTP.

In detail, the RRSTP works as follows:

1. An RRSTP switch may operates in two modes that is an Inconsistent mode and a Consistent mode. Inconsistent mode is a transient mode that lasts only few seconds. Whereas Consistent mode is an stable mode.
2. An RRSTP switch initializes Network Identifier, Sequence Number and Originator Root Path Cost fields of its Switch Priority Vector Pair to all 1s.
3. In RRSTP a received Configuration BPDU, stored in Message Priority Vector Pair, can be differentiated into five types specifically Better BPDU, Inferior BPDU, Inconsistent BPDU, Repeated BPDU and Refresher BPDU (See Figure 7)
4. In RRSTP, a received Configuration BPDU is considered as Better BPDU if
   a. Network Priority Vector of Message Priority Vector Pair is better than (numerically less than) or same as (numerically equal to) that of Port Priority Vector Pair and Configuration Priority Vector of Message Priority Vector Pair is better than (numerically less than) that of Port Priority Vector Pair.
   b. Network Priority Vector of Message Priority Vector Pair is better than (numerically less than) that of Port Priority Vector Pair and Configuration Priority Vector of Message Priority Vector Pair is worse than (numerically greater than) that of Port Priority Vector Pair but Designated Switch Identifier and Designated Port Identifier are same (numerically equal) in both Priority Vector Pairs and the received BPDU is not on the root port.
5. In RRSTP, a received Configuration BPDU is known as Inconsistent BPDU if Network Priority Vector of Message Priority Vector Pair is better than (numerically less than) that of Port Priority Vector Pair and Configuration Priority Vector of Message Priority Vector Pair is worse than (numerically greater than) that of Port Priority Vector Pair but Designated Switch Identifier and Designated Port Identifier are same (numerically equal) in both Priority Vector Pairs and the received BPDU is on the root port.
6. In RRSTP, a received Configuration BPDU is identified as Refresher BPDU if Network Priority Vector of Message Priority Vector Pair is better than (numerically less than) that of Port Priority Vector Pair but Configuration Priority Vector of Message Priority Vector Pair is same as (numerically equal to) that of Port Priority Vector Pair and the received BPDU is on the root port.

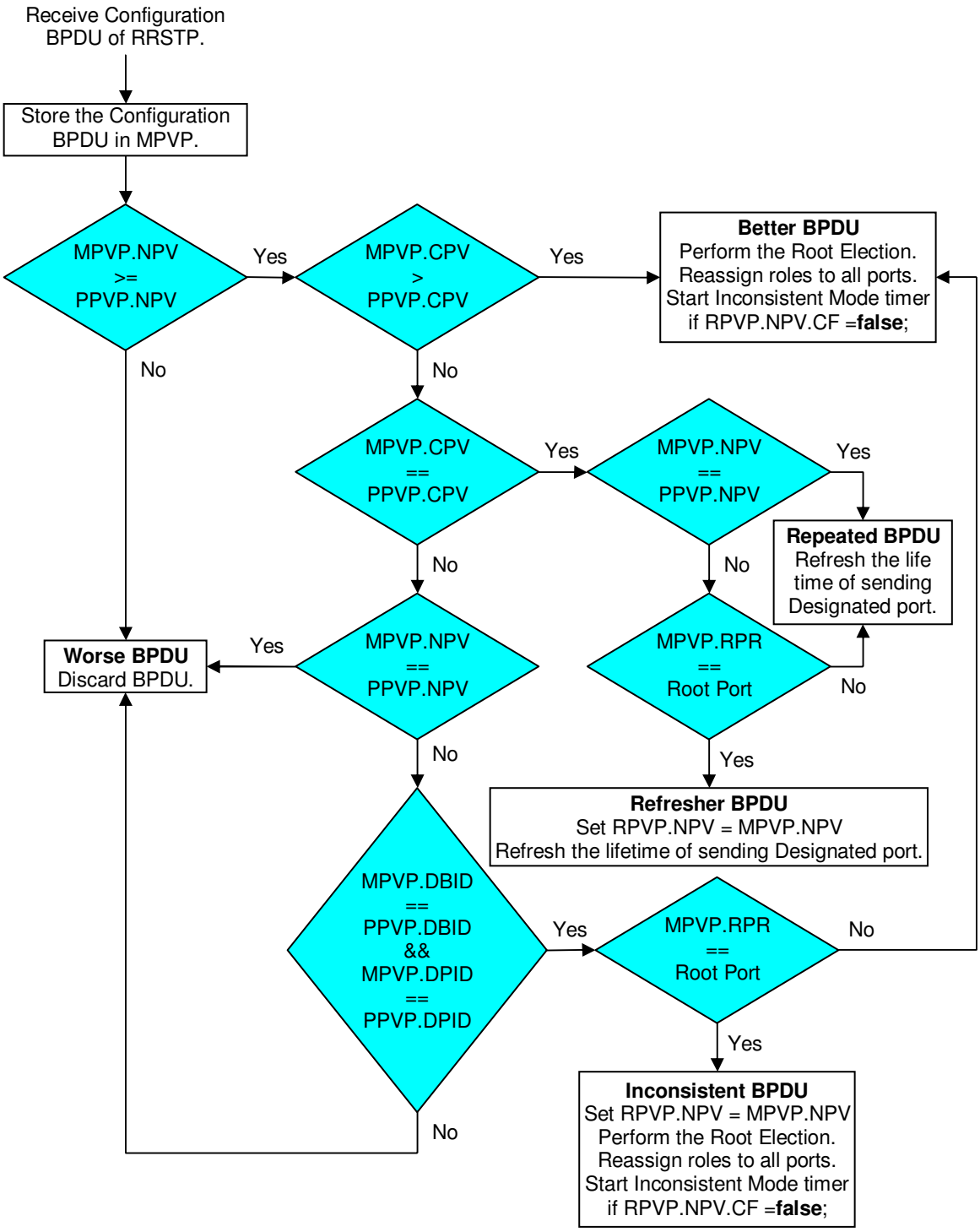


FIGURE 7: Processing of received Configuration BPDU in RRSTP.

7. In RRSTP, a received Configuration BPDU is treated a Repeated BPDU if
   a. Network Priority Vector of Message Priority Vector Pair is better than (numerically less than) that of Port Priority Vector Pair but Configuration Priority Vector of Message Priority Vector Pair is same as (numerically equal to) that of Port Priority Vector Pair and the received BPDU is not on the root port.

   b. Both Network Priority Vector and Configuration Priority Vector of Message Priority Vector Pair are same as (numerically equal to) those of Port Priority Vector Pair respectively.
8. In RRSTP, a received Configuration BPDU is recognized as Worse BPDU if
   a. Network Priority Vector of Message Priority Vector Pair is worse than (numerically greater than) that of Port Priority Vector Pair
   b. Network Priority Vector of Message Priority Vector Pair is same as (numerically equal to) that of Port Priority Vector Pair but Configuration Priority Vector of Message Priority Vector Pair is worse than (numerically greater than) that of Port Priority Vector Pair.
   c. Network Priority Vector of Message Priority Vector Pair is better than (numerically less than) that of Port Priority Vector Pair but Configuration Priority Vector of Message Priority Vector Pair is worse than (numerically greater than) that of Port Priority Vector Pair and Designated Switch Identifier and Designated Port Identifier are not same (numerically equal) in both Priority Vector Pairs.
9. An RRSTP switch always encodes the BPDU transmitting through a port with Network Priority Vector of Root Priority Vector Pair and Configuration Priority Vector of that port's Designated Priority Vector Pair.
10. An RRSTP switch when receives a Refresher BDPU, first it sets the Network Priority Vector of its Root Priority Vector Pair to that of the root port's Message Priority Vector Pair and then forces all the ports to transmit Configuration BPDUs.
11. An RRSTP switch sets the Network Priority Vector of its Root Priority Vector Pair to that of the root port's Message Priority Vector Pair, performs the Root Election and reassigns role to all the ports when it receives an Inconsistent BPDU.
12. An RRSTP switch performs the Root Election and reassigns role to all the ports if: (see Figure 8)
   a. It receives a Better BPDU
   b. An edge port has just failed or disabled.
   c. An Alternate or Backup port has just failed, disabled, aged out or suffered from a change in Port Cost or Port Identifier.
   d. A Designated port has just suffered from a change in Port Cost.
13. When RRSTP switch that Originator Root Path Cost of Network Priority Vector of Root Priority Vector Pair is worse than (numerically greater than) Root Path Cost of Configuration Priority Vector of Root Priority Vector Pair has just suffered from failure, disabling or change in Port Identifier of Designated port, the switch: (see Figure 8)
   a. Sets Originator Root Path Cost of Network Priority Vector of Root Priority Vector Pair and that of Network Priority Vector of the root port's Port Priority Vector Pair equal to Root Path Cost of Configuration Priority Vector of Root Priority Vector Pair.
   b. Sets Request Priority Vector of port equal to first four fields of Network Priority Vector of Root Priority Vector Pair.
   c. Transmits Configuration BPDU on all ports and Request BPDU on the root port.
   d. Performs the Root Election and reassigns role to all ports.
14. If an RRSTP switch that Originator Root Path Cost of Network Priority Vector of Root Priority Vector Pair is not worse than (numerically not greater than) Root Path Cost of Configuration Priority Vector of Root Priority Vector Pair has just suffered from failure, disabling or change in Port Identifier of Designated port, the switch: (see Figure 8)
   a. Sets the first three fields of Request Priority Vector of port equal to those of Network Priority Vector of Root Priority Vector Pair.
   b. Sets Originator Root Path Cost of Request Priority Vector equal to one less than that of Network Priority Vector of Root Priority Vector Pair.
   c. Transmits Request BPDU on the root port.
   d. Performs the Root Election reassigns role to all ports.
15. When an RRSTP switch that Originator Root Path Cost of Network Priority Vector of Root Priority Vector Pair is worse than (numerically greater than) Root Path Cost of Configuration Priority Vector of Root Priority Vector Pair has just suffered from failure, disabling, aging out or change in Port Cost or Port Identifier of the root port, the switch: (see Figure 8)

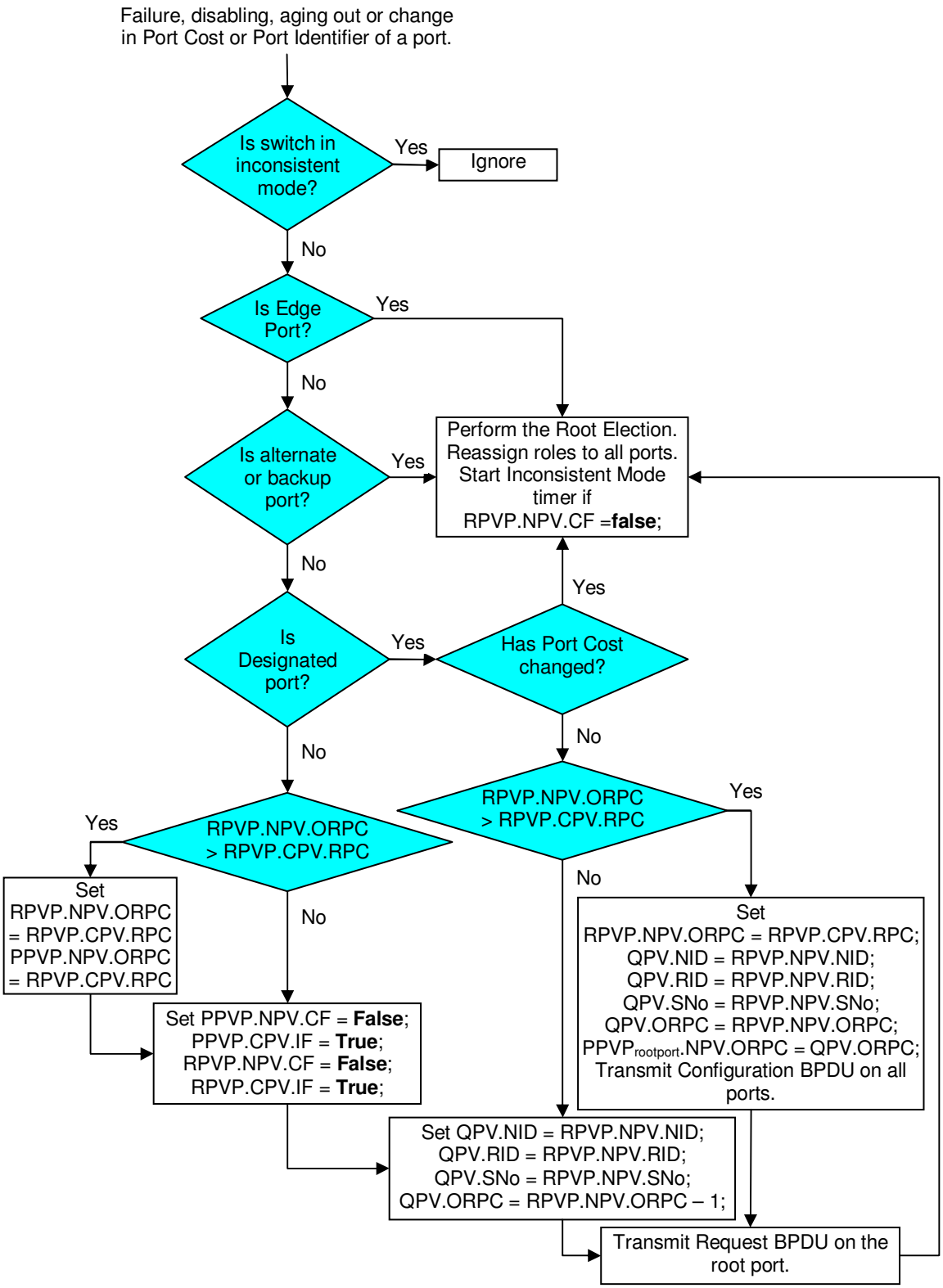


FIGURE 8: Handling of port failure and management changes on port in RRSTP

a. Sets Originator Root Path Cost of Network Priority Vector of Root Priority Vector Pair and that of Network Priority Vector of the root port's Port Priority Vector Pair equal to

Root Path Cost of Configuration Priority Vector of Root Priority Vector Pair.

b. Clears corresponding Consistent Flags and sets the corresponding Inconsistent Flags of both Root Priority Vector Pair and the root port's Port Priority Vector Pair.
c. Sets the first three fields of Request Priority Vector of port equal to those of Network Priority Vector of Root Priority Vector Pair.
d. Sets Originator Root Path Cost of Request Priority Vector equal to one less than that of Network Priority Vector of Root Priority Vector Pair.
e. Transmits Request BPDU on the root port of the switch.
f. Performs the Root Election and reassigns role to all ports.

16. If an RRSTP switch that Originator Root Path Cost of Network Priority Vector of Root Priority Vector Pair is not worse than (numerically not greater than) Root Path Cost of Configuration Priority Vector of Root Priority Vector Pair has just suffered from failure, disabling, aging out or change in Port Cost or Port Identifier of the root port, the switch: (see Figure 8)
    a. Clears corresponding Consistent Flags and sets corresponding Inconsistent Flag of Root Priority Vector Pair and the root port's Port Priority Vector Pair.
    b. Sets the first three fields of Request Priority Vector of port equal to those of Network Priority Vector of Root Priority Vector Pair.
    c. Sets Originator Root Path Cost of Request Priority Vector equal to one less than that of Network Priority Vector of Root Priority Vector Pair.
    d. Transmits Request BPDU on the root port of the switch.
    e. Performs the Root Election and reassigns role to all ports.

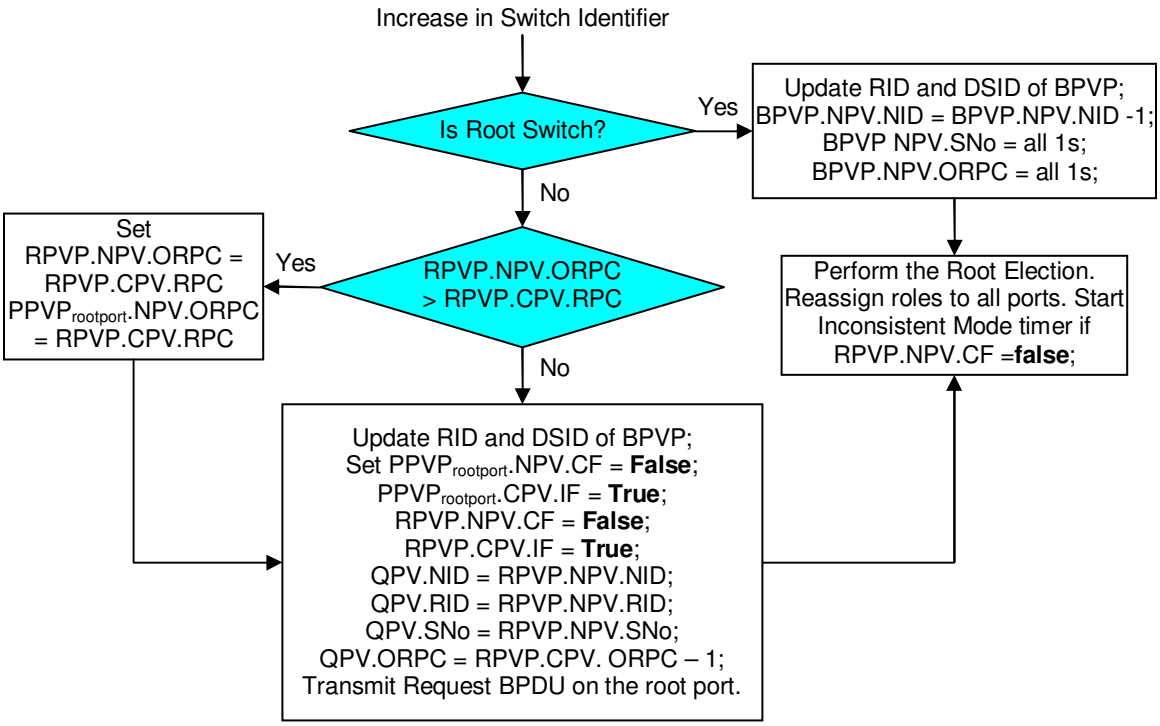


FIGURE 9: Handling of increase in Switch Identifier in RRSTP.

17. When a non-root RRSTP switch that Originator Root Path Cost of Network Priority Vector of Root Priority Vector Pair is worse than (numerically greater than) Root Path Cost of Configuration Priority Vector of Root Priority Vector Pair has just had worse Switch Identifier i.e. there is a numerical increase in Switch Identifier, the switch (See Figure 9).
    a. Sets Originator Root Path Cost of Network Priority Vector of Root Priority Vector Pair and that of Network Priority Vector of the root port's Port Priority Vector Pair equal to Root Path Cost of Configuration Priority Vector of Root Priority Vector Pair.

b. Updates Root Switch Identifier and Designated Switch Identifier of its Switch Priority Vector Pair.
c. Clears corresponding Consistent Flags and sets corresponding Inconsistent Flags of Root Priority Vector Pair and the root port's Port Priority Vector Pair.
d. Sets the first three fields of Request Priority Vector of port equal to those of Network Priority Vector of Root Priority Vector Pair.
e. Sets Originator Root Path Cost of Request Priority Vector equal to one less than that of Network Priority Vector of Root Priority Vector Pair.
f. Transmits Request BPDU on the root port of the switch.
g. Performs the Root Election and reassigns role to all ports.

18. When a non-root RRSTP switch that Originator Root Path Cost of Network Priority Vector of Root Priority Vector Pair is not worse than (numerically not greater than) Root Path Cost of Configuration Priority Vector of Root Priority Vector Pair has just had worse Switch Identifier i.e. there is a numerical increase in Switch Identifier, the switch (See Figure 9).
    a. Updates Root Switch Identifier and Designated Switch Identifier of its Switch Priority Vector Pair.
    b. Clears corresponding Consistent Flags and sets corresponding Inconsistent Flags of Root Priority Vector Pair and the root port's Port Priority Vector Pair.
    c. Sets the first three fields of Request Priority Vector of port equal to those of Network Priority Vector of Root Priority Vector Pair.
    d. Sets Originator Root Path Cost of Request Priority Vector equal to one less than that of Network Priority Vector of Root Priority Vector Pair.
    e. Transmits Request BPDU on the root port of the switch.
    f. Performs the Root Election and reassigns role to all ports.
19. When the Root RRSTP switch has just had worse Switch Identifier i.e. there is a numerical increase in Switch Identifier, then the switch.(See Figure 9)
    a. Updates Root Switch Identifier and Designated Switch Identifier of its Switch Priority Vector Pair.
    b. Decreases Network Identifier of Switch Priority Vector Pair by one.
    c. Set Sequence Number and Originator Root Path Cost of Switch Priority Vector Pair to all 1 (in binary).
    d. Performs the Root election and reassigns role to all ports.

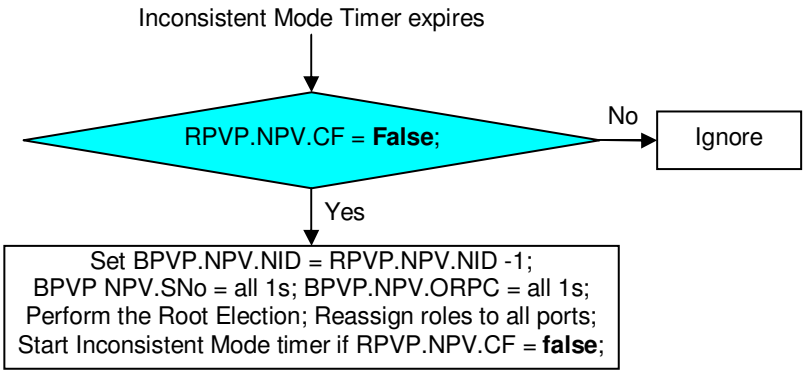


FIGURE 10: Handling of expiration of Inconsistent Mode Timer in RRSTP.

20. An RRSTP switch that is operating in Inconsistent Mode and has just suffered from expiration of Inconsistent Mode timer must: (See Figure 10)
    a. Decrease the Network Identifier of its Switch Priority Vector Pair by one.
    b. Set Sequence Number and Originator Root Path Cost of Switch Priority Vector Pair to all 1 (in binary).
    c. Perform the Root Election and reassign role to all ports.

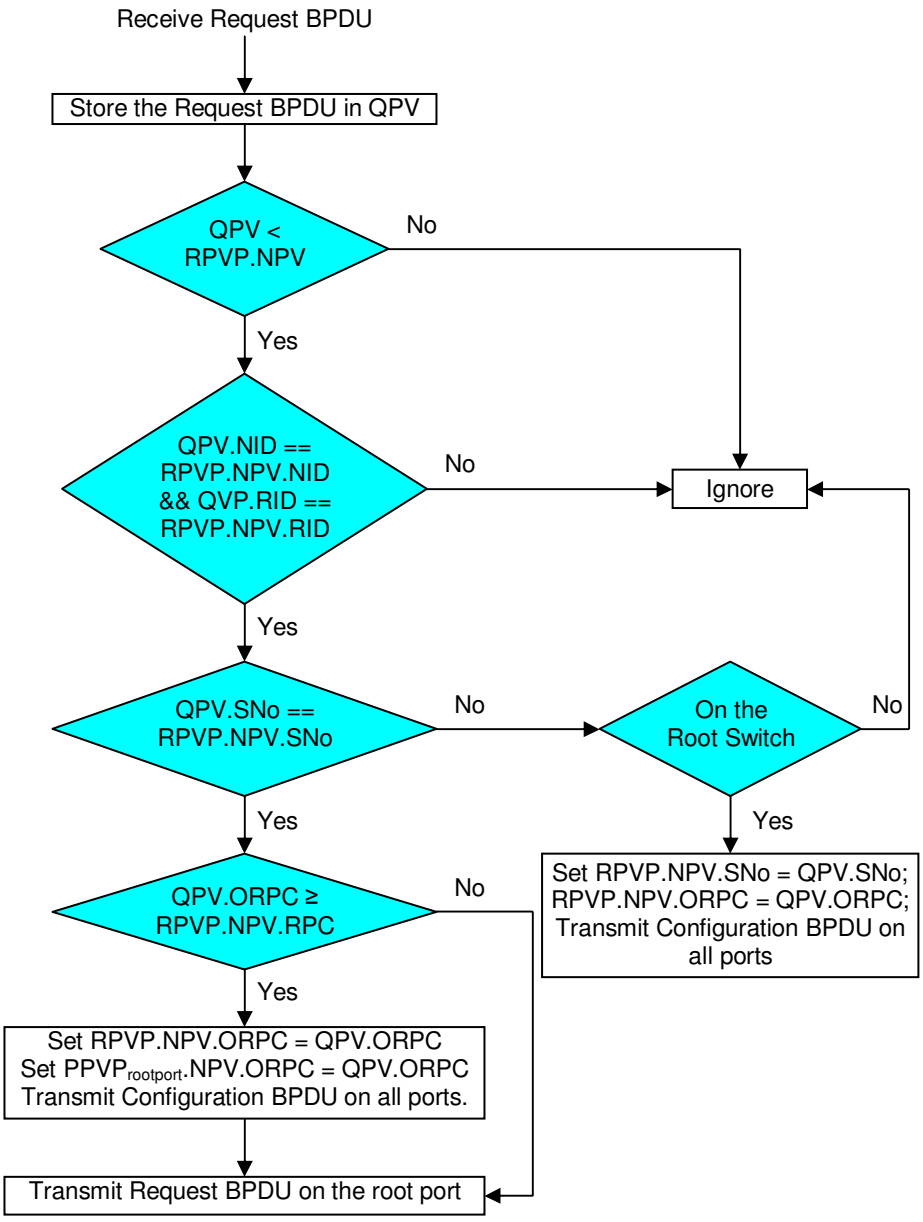


FIGURE 11: Processing of received Request BPDU in RRSTP.

21. An RRSTP switch transmits the received Request BPDU through its root port if the first three fields in the Request BPDU are same as (numerically equal to) those of Root Priority Vector Pair but Originator Root Path Cost of Request BPDU is better than (numerically less than) both Originator Root Path Cost and Root Path Cost of Root Priority Vector Pair. (see Figure 11)
22. When an RRSTP switch receives a Request BPDU such that the first three fields in the Request BPDU are same as (numerically equal to) those of Root Priority Vector Pair and Originator Root Path Cost of Request BPDU is better than (numerically less than) that of Root Priority Vector Pair but worse than or same as (numerically greater than or equal to) Root Path Cost of Configuration Priority Vector of Root Priority Vector Pair, then the switch: (see Figure 11)
    a. Sets Originator Root Path Cost of Root Priority Vector Pair to that of Request BPDU.
    b. Sets Originator Root Path Cost of the root port's Port Priority Vector Pair (or Switch Priority Vector Pair in case of the Root switch) to that of Request BPDU.

c. Transmits Configuration BPDU on all the ports and Request BPDU on the root port.

23. When the RRSTP Root Switch receives a Request BPDU such that the first two fields in the Request BPDU are same as (numerically equal to) those of Root Priority Vector Pair but Sequence Number of Request BPDU is better than (numerically less than) that of Root Priority Vector Pair, then the switch: (see Figure 11)
    a. Sets Sequence Number and Originator Root Path Cost of Root Priority Vector Pair and Switch Priority Vector Pair to that of Request BPDU.
    b. Transmits Configuration BPDU on all the ports.

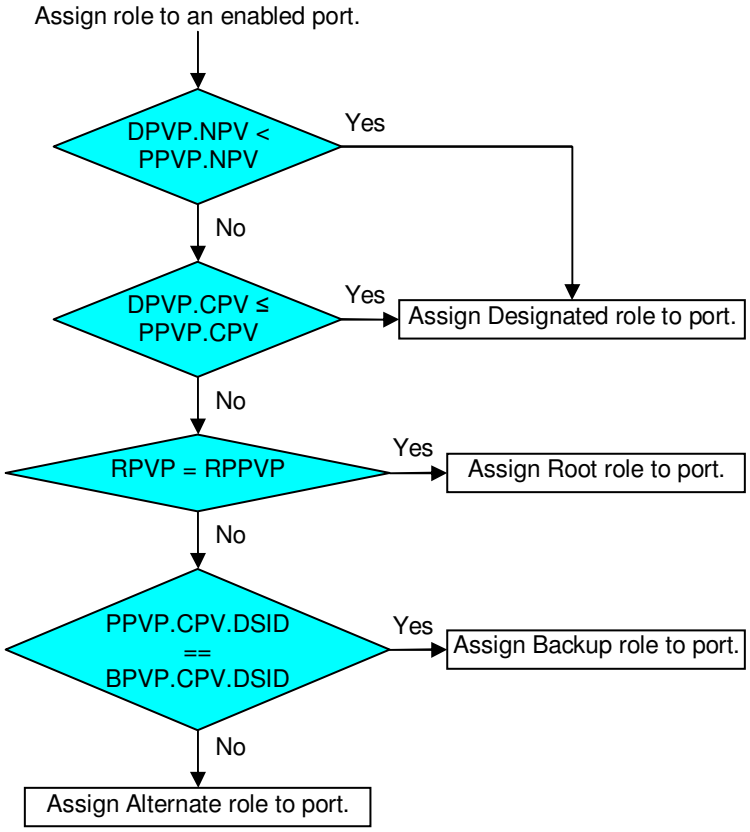


FIGURE 12: Assignment of port role to an enabled port in RRSTP.

24. An RRSTP switch assigns role to all ports, after the Root election , as follows: (See Figure 12)
    a. Assign Disabled role to a port if it is not enabled (operational).
    b. Assigns Designated role to the port if Network Priority Vector of its Designated Priority Vector Pair is better than (numerically less than) that of its Port Priority Vector Pair.
    c. Assigns Designated role to the port if Network Priority Vector of its Designated Priority Vector Pair is not better than (numerically not less than) that of its Port Priority Vector Pair but Configuration Priority Vector of its Designated Priority Vector Pair is not worse than (numerically not greater than) that of its Port Priority Vector Pair.
    d. Assigns Root role to the port if its Root Path Priority Vector Pair is same as (numerically equal to) Root Priority Vector Pair of the switch.
    e. Assigns Alternate role to the port if Network Priority Vector and Configuration Priority Vector of port's Designated Priority Vector Pair are not better than (numerically not less than) those of port's Port Priority Vector Pair, port's Root Path Priority Vector Pair is not same as (numerically not equal to) Root Priority Vector Pair of the switch and

Designated Switch Identifier of Configuration Priority Vector of port's Port Priority Vector Pair is not same as (numerically not equal to) Switch Identifier of the switch.

f. Assigns Backup role to all remaining ports.

25. An RRSTP switch forces all the ports to transmit Configuration BPDU whenever it performs the Root Election.
26. An RRSTP switch performs the Root Election as follows
    a. Find Root Path Priority Vector Pair that has the best (numerically least) Configuration Priority Vector from set of those Root Path Priority Vector Pairs that corresponding Network Priority Vectors are not worse (numerically not greater than) than that of Root Priority Vector Pair.
    b. Sets Originator Root Path Cost of Switch Priority Vector Pair to all 1 (in binary) and Network Identifier of Switch Priority Vector Pair to that of Root Path Priority Vector Pair elected in step a., if its Network Identifier is better than (numerically less than) that of Switch Priority Vector Pair.
    c. Set the Root Priority Vector Pair to that Priority Vector Pair that has better (numerically lesser) Configuration Priority Vector from set consist of Switch Priority Vector Pair and Root Path Priority Vector Pair elected in step a.

Discussion

In RSTP, a non-root switch can both generate and relay a BPDU but there is no mark distinction between them. In contrast, a non-root switch in RRSTP generates a configuration BPDU only after stamping it with its own or worse Root Path Cost i.e. setting its Originator Root Path Cost to its own or worse Root Path Cost. Hence Configuration BPDU that is relayed can be distinguished easily from one that is generated by a non-root switch in RRSTP.

RSTP is vulnerable to count-to-infinity because an RSTP switch suffering from a recent root port failure or management change on the root port may use a stale RST BPDU generated by a switch using its orphan alternate port either as the root port or designated port. In contrast, an RRSTP switch can distinguish and discard such problematic BPDUs. To obtain it, first an RRSTP switch processes only those Configuration BPDUs that have either better (numerically smaller) Sequence Number than the switch or have same Sequence Number but better Originator Root Path Cost than the switch. Second, an RRSTP switch suffering from a recent root port failure or a management change on the root port always sets its Originator Root Path Cost to its own Root Path Cost or better and clears its Consistent Flag (see Figure 8). The above two steps ensures that a switch suffering from a recent root port failure or a management change on the root port will not process a Configuration BPDU generated by a switch in the orphan subtree. As a switch in orphan subtree may generate a Configuration BPDU with Originator Root Path Cost at most equal to its own Root Path Cost which is always worse than that of a switch suffering from a recent root port failure or a management change on the root port.

As alternate ports are not using by RRSTP to restore the network connectivity, RRSTP uses the novel Rapid BPDU Distribution Mechanism in order to facilitate switches for quick and rapid convergence. Figure 8 is showing the procedure of generating a Request BPDU on the root port by a switch when one of its designated ports fails or suffers from a management change. Where as Figure 11 is illustrating the generation of Configuration BPDUs by a switch when it receives a Request BPDU that has Originator Root Path Cost worse than that switch's Root Path Cost.

Rapid BPDU Distribution Mechanism is effective only in temporary segregation of network. In case of absolute segregation of network, RRSTP reelects the root switch after expiration of Inconsistent Mode timer. RRSTP uses a conservative value (say 6 seconds) for Inconsistent Mode timer in order to provide enough time for Rapid BPDU Distribution Mechanism to successfully transmit Configuration BPDUs. It can effort such a conservative value because a switch running Inconsistent Mode timer remains intact with its previous topology and so a stations in the orphan subtree can communicate with other stations in the subtree.

Interoperability With Legacy Switches

RRSTP is not backward compatibility with legacy STP and RSTP switches. However, RRSTP can be integrated with DRSTP [5], a backward compatible solution to reasonably improve the reliability of legacy Ethernet networks. So, an DRSTP integrated RRSTP switch must operates in two different phases explicitly an DRSTP phase and an RRSTP phase. An DRSTP integrated RRSTP switch will said to be in DRSTP phase if the root port of the switch is receiving STP Configuration BPDU, RST BPDU or DRST BPDU. Otherwise the switch is said to be in RRSTP phase. When an DRSTP integrated RRSTP switch is in DRSTP phase, a port can operate as STP, RSTP or DRSTP port depending upon other switches on the link connected to that port. However, in RRSTP phase, a switch may have both legacy ports (STP, RSTP or DRSTP ports) and RRSTP ports. RRSTP Configuration BPDU must be converted into legacy BPDU, by simply stripping off newly added fields, before transmitting on a legacy port. Moreover, Root Election in an DRSTP integrated RRSTP switch is a bit more complex than that in RRSTP switch. First, the switch drives the DRSTP Root Priority Vector, using the DRSTP rules, from set of legacy ports (STP, RSTP or DRSTP ports). Second, the switch drives the RRSTP Root Priority Vector Pair, using the RRSTP rules, from set of RRSTP ports. Third, the switch drives stripped Configuration Vector. It is nothing but the Configuration Priority Vector of RRSTP Root Priority Vector Pair such that all the newly add fields of RRSTP are stripped off. The switch then moves into DRSTP phase and starts using its DRSTP Root Priority Vector if its DRSTP Root Priority Vector is better than stripped Configuration Vector. Otherwise the switch moves into RRSTP phase and starts using its RRSTP Root Priority Vector Pair.

Comparison with Contemporary Protocols

This section will critically discuss RRSTP with other contemporary protocols. The four other protocols that will be used for comparison are STP [11], RSTP [1], DRSTP [5] and RSTP with Epoch [3][4]. The five key aspects that will be discussed during comparison are vulnerability against count-to-infinity, convergence time, scalability, protocol implementation and backward compatibility.

Both STP [11] and RSTP [1] are susceptible to both temporary and absolute count-to-infinities as they cannot distinguish between stale and fresh BPDUs. In contrast, DRSTP [5] provide protection, to some extent, against both type of count-to-infinities. It is achieved by inserting a small delay to avoid usage of probably stale information. “RSTP with Epoch” [3][4] is a new protocol that is specifically designed to address the count-to-infinity problem but unfortunately it is vulnerable against temporary count-to-infinity. It is because a new epoch is stated in this protocol only when the switch suffering from root port failure has no alternate port. Fortunately, RRSTP is not vulnerable to both temporary and absolute count-to-infinities as it can distinguish between fresh and stale BPDUs and it also does not use orphan alternate ports to restore connectivity.

STP exhibits very slow convergence time of up to 50s [12] due to use of conservative timers. This also makes STP a low scalable protocol. In contrast, RSTP may converge with in 1-3s due to its aggressive and optimistic approach. But this low convergence time is showed by RSTP only in the absence of count-to-infinity. This vulnerability of RSTP also adversely affects its scalability. On the other hand, DRSTP usually exhibits convergence time of 1-3s as it can prevent count-to-infinity in most cases. Moreover, the scalability of DRSTP is generally more than RSTP. Convergence time of “RRSTP with Epoch” is order of round trip time of BPDU to the Root Switch [3][4]. In contrast, convergence time RRSTP is order of trip of BPDU around the shortest broken cycle. Further, scalability “RSTP with Epoch” is under question due to its convergence process. This protocol starts a new epoch and so reelection of the Root Switch whenever a switch having no alternate port experiences the failure of its root port. This reelection of the Root Switch produces an unnecessary network wide disturbance even when it is possible to reconverge the network without such reelection. In contrast, RRSTP avoids the reelection of the Root Switch until the expiration of Inconsistent Mode Timer. Moreover, it uses its highly decentralized “Rapid BPDU Distribution Mechanism” to reconverge the network. Hence, it expected that RRSTP will be much more scalable as compare to “RSTP with Epoch”.

The functionality of RRSTP is also very much similar to DSDV [13]. But, two major aspects in which RRSTP differ from DSDV are getting rid of settling timer and use of its decentralized "Rapid BPDU Distribution Mechanism". These two aspects help RRSTP to keep its convergence time as low as possible.

RSTP [1] is backward compatible to STP [11]. DRSTP [5] is also backward compatible to STP [11] and RSTP [1]. But it cannot prevent count-to-infinity in the presence of STP switches in the network. Similarly, "RSTP with Epoch" [3][4] is also backward compatible to STP [11] and RSTP.[1] But this compatibility is provided at the cost of exposure of network to count-to-infinity. RRSTP can be also made backward compatible by integrating it with DRSTP [5].

In a nutshell, RRSTP is much superior to its contemporary protocol in most major aspects of network. Table 1 is showing the comparison of RRSTP with other contemporary protocols in a tabular form.

		STP	RSTP	DRSTP	RSTP with Epoch	RRSTP
Frequency of Count-to-infinity	Temporary	High	High	Low	High	Zero
	Absolute	High	High	Low	Zero	Zero
Convergence time	In case of no count-to-infinity	Up to 50s	1-3s	1-3s	Order of round trip time to Root Switch	Order of round trip time around shortest broken cycle
	In case of count-to-infinity	Order of maximum message age	Order of maximum message age.	Order of maximum message age.	Order of maximum message age	N/A
Scalability		Very Low	Low	Medium	High	Very High
Backward compatibility		N/A	Yes	Yes	Yes	Yes

TABLE1: Comparison of RRSRP with other contemporary protocols.

5. RELATED WORK

There are two schools of thought to increase reliability and scalability of Ethernet. Researchers in one school of thought are suggesting to replace current spanning tree protocol with other techniques. For example Perlman proposed Rbridges [14] and Garcia et al. proposed LSOM [15] to substitute spanning tree with more reliable and scalable link state routing. Turn prohibition technique can also be used in place of legacy spanning tree protocols to improve reliability and scalability. Up/Down routing proposed by Shoreder et al. [16], Turn Prohibition (TB) proposed by Starobinski et al. [17], Tree-Based Turn-Prohibition (TBTP) proposed by Pellegrini et al. [18] and Hierarchal Up/Down Routing and Bridging Architecture (HURP/HURBA) proposed by Ibáñez et al. [19] are few well-known algorithms based on this technique.

SEATTLE proposed by Kim et al. [20] is a completely new layer 2 network architecture. However, it is not backward compatible solution. Sharma et al. [21] introduce a multiple spanning tree architecture that improves the throughput and reliability over when using a single spanning tree. SmartBridges [22] uses the techniques of diffusing computation [23] and effective global consistency to achieve loop-freeness.

There are researcher in another school of thought that believe that reliability and scalability of spanning tree protocols can be enhance by making few modification in its operation. First serious attempt, to the best of my knowledge, is made by Elmeleegy et al. by proposing "RSTP with Epochs" [3],[4]. Other protocols that address the reliability of spanning tree protocol are DRSTP

[5] and Ether Fuse [24] proposed by Atif and Elmeleegy et al. respectively DRSTP was proposed to protect legacy RSTP controlled Ethernet networks against count-to-infinity to some extent. Ether Fuse [24] is a hardware device acting like a fuse that burn out logically before the count-to-infinity problem occurring in the network become severe.

6. CONCLUSION

This paper presents a novel spanning tree protocol, RRSTP, to safeguard the underlying Ethernet network against highly undesirable count-to-infinity problem. The protocol makes subtle modification in both structure and processing of BPDUs to achieve this goal. Further, to make the underlying Ethernet network highly available, RRSTP uses a novel decentralized "Rapid BPDU Distribution Mechanism. With the help of this mechanism, RRSTP can converge the network in order of time required for BPDU to travel along the shortest broken cycle. In a nutshell, RRSTP is expected to out class all its contemporary spanning tree protocols in all three key features specifically reliability, scalability and availability. Thus, Ethernet can now safely used along with RRSTP even in highly mission critical networks.

7. REFERENCES

1. LAN/MAN Standards Committee of the IEEE Computer Society. *"IEEE Standard for Local and metropolitan area networks: Media Access Control (MAC) Bridges - 802.1D"*. 2004.

2. Myers, T. E. Ng, and H. Zhang. *"Rethinking the Service Model: Scaling Ethernet to a Million Nodes"*. In Proceedings of the 3rd Workshop on Hot Topics in networks, San Diego,.CA, USA, 2004.

3. K. Elmeleegy, A. L. Cox and T. S. E. Ng. *"On Count-to-Infinity Induced Forwarding Loops in Ethernet Networks"*. In Proceedings of the 25th IEEE Infocom, Barcelona, Catalunya, Spain, 2006.

4. K. Elmeleegy, A. L. Cox and T. S. E. Ng. *"Understanding and Mitigating the Effects of Count to Infinity in Ethernet Networks"*. IEEE/ACM Transactions on Networking, 17(1):186-199, 2009.

5. S M Atif. *"DRSTP: A Simple Technique for Preventing Count-to-Infinity in RSTP Controlled Switched Ethernet Networks"*. International Journal of Computer Networks, 2(6):278-296, 2011.
6. R. Perlman. "*An Algorithm for Distributed Computation of a Spanning Tree in an Extended LAN"*. In the proceedings of 9th ACM Data Communications Symposium. New York, USA, 1985.

7. M Seaman. *"High Availability Spanning Tree"*. [online] Available at: www.ieee802.org/1/files/public/docs1998/hasten7.pdf. [Accessed 21 March 2011].

8. M. Seaman. *"Speedy Tree Protocol"*. [online] Available at: www.ieee802.org/1/files/public/docs1999/speedy_tree_protocol_10.pdf. [Accessed 21 March 2011].

9. M. Seaman. *"Truncating Tree Timers"*. [online] Available at: www.ieee802.org/1/files/public/docs1999/truncating_tree_timing_10.pdf. [Accessed 21 March 2011].

10. V. Jain and M. Seaman. *"Faster flushing with fewer addresses"*. [online] Available at: www.ieee802.org/1/files/public/docs1999/fast_flush_10.pdf. [Accessed 21 March 2011].

11. LAN/MAN Standards Committee of the IEEE Computer Society. *"IEEE Standard for Information technology – Telecommunications and information exchange between systems –*

Local and metropolitan area networks – common specifications, Part 3: Media Access Control (MAC) Bridges" , ISO/IEC 15802-3, ANSI/IEEE Std 802.1D, 1998.

12. Cisco Systems, Inc. *"Spanning Tree Protocol Problems and Related Design Considerations"*. [online] Available at: www.cisco.com/en/US/tech/tk389/tk621/technologies_tech_note09186a00800951ac.shtml [Accessed 21 March 2011].

13. C. E. Perkins and P. Bhagwat. *"Highly dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers"*. In Proceedings of the ACM SIGCOMM 1994, London, UK, 1994.

14. R. Perlman. *"Rbridges: Transparent routing"*. In Proceedings of the 23rd IEEE Infocom, Hong Kong, 2004.

15. R. Garcia, J. Duato and F. Silla. *"LSOM: A link state protocol over MAC addresses for metropolitan backbones using optical Ethernet switches"*. In Proceedings of the 2nd IEEE International Symposium on Network Computing and Applications, Cambridge, MA, USA, 2003.

16. M. Schroeder, A. Birrell, M. Burrows, H. Murray, R. Needham, T. Rodeheffer, E. Satterthwaite, C. Thacker. *"Autonet: A High-Speed, Self–Configuring Local Area Network Using Point–to–Point Links"*. IEEE Journal on Selected Areas in Communications, 9(8):1318–1335, 1991.

17. D. Starobinski, G. Karpovsky, F. Zakrevsky. *"Applications of network calculus to general topologies"*, IEEE/ACM Transactions on Networking, 11(3):411–422, 2003.

18. F. D. Pellegrini, D. Starobinski, M. G. Karpovsky and L. B. Levitin. *"Scalable cycle-breaking algorithms for gigabit Ethernet backbones"*. In Proceedings of the 23rd IEEE Infocom, Hong Kong, 2004.

19. Guillermo Ibáñez, Alberto García-Martínez, Juan A. Carral, Pedro A. González, Arturo Azcorra, José M. Arco. *"HURP/HURBA: Zero-configuration hierarchical Up/Down routing and bridging architecture for Ethernet backbones and campus networks"*, Computer Networks, 54(1):41-56,2010.

20. C. Kim, M. Caesar, and J. Rexford. "*Floodless in SEATTLE: A Scalable Ethernet Architecture for Large Enterprises"*. In Proceedings of the ACM SIGCOMM. 2008, Seattle, WA, USA, 2008.

21. S. Sharma, K. Gopalan, S. Nanda, and T. Chiueh. "*Viking: A multispanning tree Ethernet architecture for metropolitan area and cluster networks"*. In Proceedings of the 23rd IEEE Infocom, Hong Kong,. 2004.

22. T. L. Rodeheffer, C. A. Thekkath, and D. C. Anderson. *"SmartBridge: A scalable bridge architecture"*. In Proceedings of the ACM SIGCOMM. 2000, Stockholm, Sweden, 2000.

23. E. W. Dijkstra, C. S. Scholten. *"Termination detection for diffusing computations"*. Information Processing Letters, 11(1):1-4, 1980.

24. K. Elmeleegy, A. L. Cox and T. S. E. Ng. *"EtherFuse: An Ethernet Watchdog"*. In Proceedings of the ACM SIGCOMM 2007, Kyoto, Japan, 2007.

Fuzzy Controller Based Stable Routes with Lifetime Prediction in MANETs

Taqwa Odey taqwa.odey@gmail.com
Department of Computers Engineering
University of Basra
Basra,Iraq

Prof. Dr. Abduladhem A. Ali abduladem1@yahoo.com
Department of Computers Engineering
University of Basra
Basra,Iraq

Abstract

In ad hoc networks, the nodes are dynamically and arbitrary located in a manner that the interconnections between nodes are changing frequently. Thus, designing an effective routing protocol is a critical issue. In this paper, we propose a fuzzy based routing method that selects the most stable route (FSRS) considering the number of intermediate nodes, packet queue occupancy, and internodes distances. Also it takes the produced cost of the selected route as an input to another fuzzy controller predicts its lifetime (FRLP), the evaluation of the proposed method is performed using OMNet++4.0 simulator in terms of packet delivery ratio, average end-to-end delay and normalized routing load.

Keywords: MANET, AODV, Fuzzy Controller, Stable Route, Route Lifetime.

1. INTRODUCTION

An ad hoc network [1] is a particular type of wireless network, where a collection of nodes forming a temporary network without the aid of any established infrastructure, or support of any centralized administration such as a base station or an access point. Each node in such network behaves not only as a host but also as a router and takes part in discovery and maintenance of routes to other nodes.

Many routing protocols for MANETs have been introduced, the efforts in this area is that routing in such networks is a significant challenging task due to the frequently changing of the network topology.

To minimize link breakage and keeping the active routes lifetime, it is important to select the most stable route. Link stability indicates how stable the link is and how long it can support communication between two nodes; it basically depends on the distance between mobile nodes and buffer zone effect [2][3].

The route lifetime is the time for which the route is considered to be valid, too long route lifetime may leads to consider some routes as valid while they were broken. In contrast, too short lifetime may leads to remove some valid routes.
Due to the uncertainty nature of the node mobility and the estimation of link breakage, fuzzy logic has been applied to reduce the effect of these problems and improve the network performance. In the following section, we present some recently proposed methods concerning this area of research:

In A. Banerjee et.al. [4] method, a fuzzy controller named (RE) is embedded in every node to evaluate the performance of a link depending on residual energy ratio, neighbor affinity, and congestion if it was an intermediate node along the path to the destination, and if it was the destination it measures the performance of the last link then combines the performance of all

the links and the hop count of that route to measure its performance to find the suitable route within a threshold time period.

While the method of S.H. Nasiri et.al. [5] Considers fuzzy nodes randomly distributed in the network, each has four inputs: number of neighbors, mobility factor of that node and its neighbors, and angle of movement to predict the link lifetime to be used by shortest path routing algorithm in selecting the next hop of the path.

E. Natsheh et.al. [6] proposed three methods to obtain fuzzy active route timeout in AODV routing protocol, where the fuzzy inputs in the first method was the number of hop count and number of control packets between two sampling interval, number of hop count and transmission power in the second method, while the last method takes the average of the results obtained from the previous two methods.

In this paper, we introduce a new method that merges these two ideas of using fuzzy controller to estimates the route cost and prediction of the route lifetime using fuzzy controller. The proposed method modifies the AODV routing protocol [7] within two stages, the first is called Fuzzy Stable Route Selection (FSRS) which selects the most stable available route based on fuzzy cost produced from three parameters: number of intermediate nodes, packet queue occupancy, and internodes distances in such a way [8] that reduces the routing discovery wait time by making each node along the path from the source to the destination participating in selecting the optimal route. While the second stage called Fuzzy Route Lifetime Prediction (FRLP) which predicts each route lifetime based on its fuzzy cost.

2. PROPOSED APPROACH

Our scheme incorporates two fuzzy controllers into each node, fuzzy controller <1> has three input metric to produce the path cost from the source to this node:

- Number of intermediate nodes
  The most popular metric used for selecting the route, since small number of intermediate nodes will have few chance of path breaking.

- Packet queue occupancy
  Routes not congested are more reliable, stable, and faster.

$$PQO = \sum_{hcnt=1}^{n} \frac{currentPQlength}{\max PQlength} \quad \ldots(1)$$

Where n denotes the number of nodes in that path, PQ denotes packet queue.

- Internodes distances

Short distance between nodes leads to high received signal strength, also the path with nearest neighbors are more stable since a small movement of any node located on the edge of other node's transmission may cause breaking in paths with far neighbors.

$$IND = \sum_{hcnt=1}^{n} \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \quad \ldots(2)$$

Where ($x_{i,j}$,$y_{i,j}$) are the x,y coordinates of node i and its previous node j .
While the fuzzy controller <2> has only single input-single output which predicts the lifetime of the selected route from the source to the destination.

During the route discovery process of the AODV routing protocol, the Route Request (RReq) message carries the sum of the input parameters of fuzzy controller <1> in their entries, when a node along the path from the source to the destination receives this RReq msg. it will work as Fuzzy Stable Route Selection (FSRS):

- Measure the required parameters at the node itself and add them to the contents of the corresponding entries of the RReq msg. before forwarding it.
- Take these entries contents as inputs to the fuzzy controller <1> to produce the fuzzy cost of this individual path.
- Compare this fuzzy cost with the previous minimum fuzzy cost, if it is smaller it will save it for next comparisons and update the reverse route entry to the source of the RReq msg. with the address of the previous node.

This process will continue until getting the destination which sends the Route Reply (RRep) message on the generated route after predicting its lifetime (FRLP) by taken its fuzzy cost as input to fuzzy controller <2>.

3. SIMULATION MODEL

The simulation modeled a network in 700 m × 700 m area with 20 / 30 mobile nodes. Each node had a channel capacity of 54 Mbps. The IEEE 802.11g was used as the medium access control protocol. A random waypoint mobility model was employed with a speed ranging from 0 to 10 m/s. Eight mobile nodes acted as traffic sources generating data packets at a rate from 2 to 4 packets/sec, and the data traffic was generated using CBR (Constant Bit Rate), UDP application, each packet size was 512 bytes. The simulation was executed for 250 seconds of simulation time by OMNeT++ 4.0 simulator [9].

4. SIMULATION RESULTS

We compare the performance of AODV routing protocol, AODV routing protocol with FSRS, AODV with FSRS and FRLP in terms of:

- ***Packet Delivery Ratio*** :

$$PDR = \frac{\sum no.of\ rcvd.data\ pkts}{\sum no.of\ sent\ data\ pkts} \quad ...(3)$$

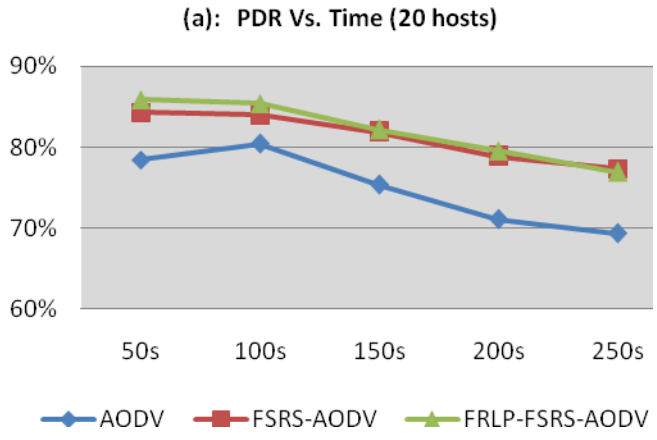
- ***Average End-to-End Delay*:**

average time taken by data packets when released by sources until reach their destinations.

- ***Normalized Routing Load*** :

$$NRL = \frac{\sum no.of\ sent\ ctr.\ pkts}{\sum no.of\ sent\ data\ pkts} \quad ...(4)$$

From the simulation results in figures (1-3), the FSRS-AODV routing overcomes the performance of the original AODV routing, about 6.087%, 44.36%, and 10.113% improving in packet delivery ratio, End-to-End delay, and normalized routing load respectively, because of its ability to select more stable, less congested, and few failures routes. While more improvement in the network performance will be obtained when using FRLP-FSRS-AODV routing, about 7.18%, 53.247%, and 22.499% improving in packet delivery ratio, End-to-End delay, and normalized routing load respectively, due to less route errors and unnecessary route discoveries resulting in reducing control traffics, routing delay, and increasing packet delivery ratio.



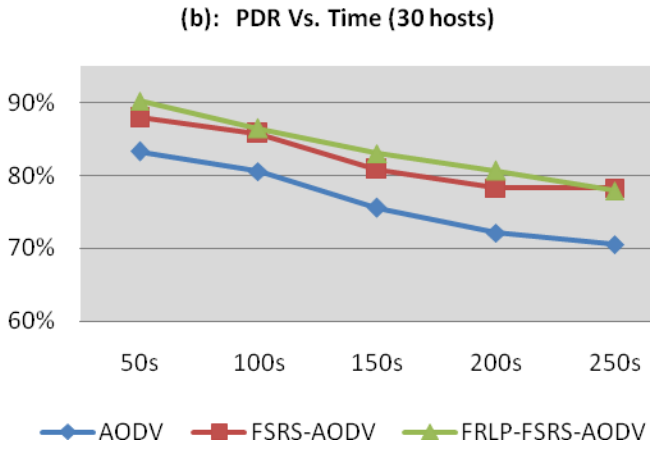
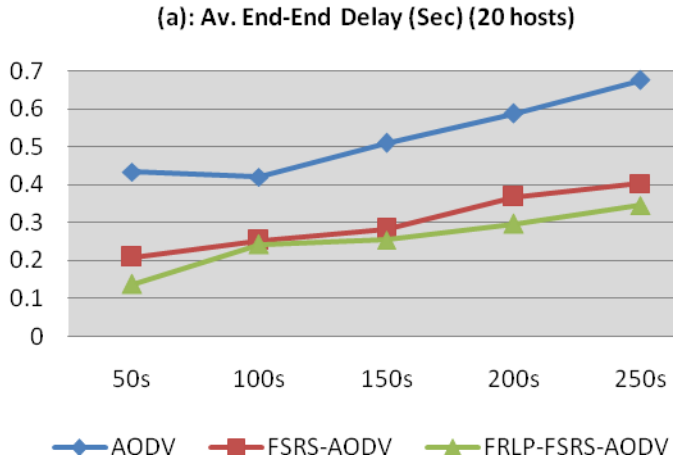


FIGURE (1): Packet Delivery Ratio vs. Time



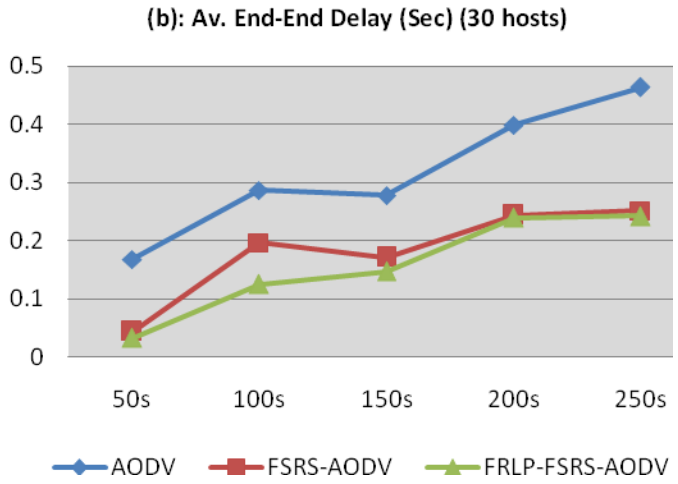
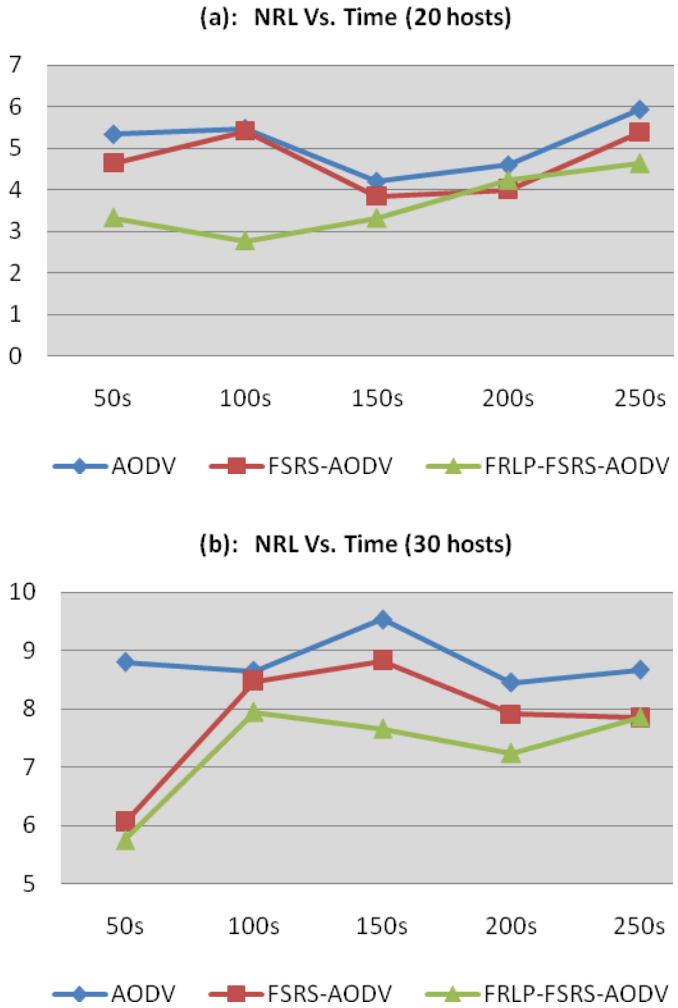


FIGURE (2): Average End-to-End Delay (Sec) Vs. Time



FIGURE(3): Normalized Routing Load vs. Time

5. CONCLUSION

In this paper, we used a fuzzy controller to obtain the routes cost depending on the number of intermediate nodes, packet queue occupancy, and internodes distances. And utilized this fuzzy cost to predict the lifetime of the selected routes using another fuzzy controller. The simulation results show that the proposed FSRS-AODV routing enhances the packet delivery ratio, average end-to-end delay, and normalized routing load when compared with the original AODV routing protocol indicating the stability of the selected routes. While more improvement will be obtained when adding FRLP indicating the suitable prediction of the selected routes lifetimes. Future work studies could take the impact of nodes mobility information that may improve the proposed method.

6. REFERENCES

1. P. Mohabatra and S. V. Krishnamorthy, "Ad Hoc Networks: Technologies and Protocols", Springer Science + Business Media Inc., 2005.

2. G. Lim, K. Shin, S. Lee, H. Yoon, and J. S. Ma, "Link Stability and Route Lifetime in Ad-hoc Wireless Networks", In the Proceedings of 1st International Conference on Parallel Processing Workshops (ICPPW'02), Vancouver, Canada, pp. 116-123, 2002.

3. D. Kumar, A. A. Kherani, and E. Altman, "Route Lifetime Based Interactive Routing In Intervehicle Mobile Ad Hoc Networks", INRIA Research Report No. RR-5691, Sophia Antipolis, France, September 2005.

4. Banerjee and P. Dutta, "Fuzzy-Controlled Route Discovery For Mobile Ad Hoc Networks", International Journal of Engineering Science and Technology, 2 (6):2347-2353, 2010.

5. S. H. Nasiri, M. Fathy, E. Z. Khosrafi, and A. Shojaeifard, "Improving Link Reliability in Mobile Ad Hoc Networks Using Fuzzy Nodes", In Proceedings of the 2008 International Conference of Communications & Information Technology, Marathon Beach, Attica, Greece, pp. 252-255, June 1-3, 2008.

6. E. Natsheh, S. Khatun, A. B. Jantan and S. Subramaniam, "Fuzzy Metric Approach For Route Lifetime Determination In Wireless Ad Hoc Networks", International journal of Ad Hoc and Ubiquitous Computing, 3 (1):1-9, 2008.

7. Parkins, E. Royer, and S. Das, " Ad hoc On-Demand Distance Vector (AODV) Routing", RFC 3561, July 2003.

8. T. O. Fahad and A. A. Ali, "Improvement of AODV Routing on MANETs Using Fuzzy Systems", In Proceedings of 1st International Conference of Energy, Power, and Control, Basra, Iraq, pp. 300-304, 30 Nov. to 2 Dec. 2010.

9. OMNeT++, http://www.omnetpp.org.

INSTRUCTIONS TO CONTRIBUTORS

The International Journal of Computer Networks (IJCN) is an archival, bimonthly journal committed to the timely publications of peer-reviewed and original papers that advance the state-of-the-art and practical applications of computer networks. It provides a publication vehicle for complete coverage of all topics of interest to network professionals and brings to its readers the latest and most important findings in computer networks.

To build its International reputation, we are disseminating the publication information through Google Books, Google Scholar, Directory of Open Access Journals (DOAJ), Open J Gate, ScientificCommons, Docstoc and many more. Our International Editors are working on establishing ISI listing and a good impact factor for IJCN.

The initial efforts helped to shape the editorial policy and to sharpen the focus of the journal. Starting with volume 3 2011, IJCN appears in more focused issues. Besides normal publications, IJCN intend to organized special issues on more focused topics. Each special issue will have a designated editor (editors) – either member of the editorial board or another recognized specialist in the respective field.

We are open to contributions, proposals for any topic as well as for editors and reviewers. We understand that it is through the effort of volunteers that CSC Journals continues to grow and flourish.

IJCN LIST OF TOPICS

The realm of International Journal of Computer Networks (IJCN) extends, but not limited, to the following:

- Algorithms, Systems and Applications
- ATM Networks
- Cellular Networks
- Congestion and Flow Control
- Delay Tolerant Networks
- Information Theory
- Metropolitan Area Networks
- Mobile Computing
- Multicast and Broadcast Networks
- Network Architectures and Protocols
- Network Modeling and Performance Analysis Network
- Network Security and Privacy
- Optical Networks
- Personal Area Networks
- Telecommunication Networks
- Ubiquitous Computing
- Wide Area Networks
- Wireless Mesh Networks
- Ad-hoc Wireless Networks
- Body Sensor Networks
- Cognitive Radio Networks
- Cooperative Networks
- Fault Tolerant Networks
- Local Area Networks
- MIMO Networks
- Mobile Satellite Networks
- Multimedia Networks
- Network Coding
- Network Operation and Management
- Network Services and Applications
- Peer-to-Peer Networks
- Switching and Routing
- Trust Worth Computing
- Web-based Services
- Wireless Local Area Networks
- Wireless Sensor Networks

CALL FOR PAPERS

Volume: 3 - **Issue:** 3 - May 2011

i. Paper Submission: May 31, 2011 **ii. Author Notification:** July 01, 2011

iii. Issue Publication: July /August 2011

CONTACT INFORMATION

Computer Science Journals Sdn BhD
M-3-19, Plaza Damas Sri Hartamas
50480, Kuala Lumpur MALAYSIA

Phone: 006 03 6207 1607
006 03 2782 6991

Fax: 006 03 6207 1697

Email: cscpress@cscjournals.org

CSC PUBLISHERS © 2011
COMPUTER SCIENCE JOURNALS SDN BHD
M-3-19, PLAZA DAMAS
SRI HARTAMAS
50480, KUALA LUMPUR
MALAYSIA

PHONE: 006 03 6207 1607
006 03 2782 6991

FAX: 006 03 6207 1697
EMAIL: cscpress@cscjournals.org