IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 4, No 3, July 2010
ISSN (Online): 1694-0784
ISSN (Print): 1694-0814

An Effective Solution to Reduce Count-to-Infinity Problem in Ethernet

[1]Ganesh.D, [2] Venkata Rama Prasad Vaddella

[1]Assistant Professor,
Department of Information Technology
Sree Vidyanikethan Engineering College, Tirupati-517 102, India

[2] Professor and Head
Department of Information Technology
Sree Vidyanikethan Engineering College, Tirupati-517 102, India

Abstract

Ethernet is currently the most popular networking technology because of its high performance, low cost and ubiquity nature. Ethernets rely on the dynamic computation of a cycle-free active forwarding topology. For that it uses rapid spanning tree protocol. Unfortunately, it exhibits count-to-infinity problem that may lead to forwarding loops under certain network failures. These consequences are considered serious since network can become highly congested and even packet forwarding can fail. In this work, a simple and effective solution to reduce count-to-infinity problem, called RSTP with epochs is proposed. This eliminates the count-to-infinity induced forwarding loop and improves the convergence time.

Keywords**:** *count-to-infinity, Ethernet, spanning tree protocols*

1. Introduction

Ethernet is the most popular networking technology in a wide range of environments due to its high performance-to-cost ratio and ubiquity. In existing ethernet standards, packet flooding is used to deliver a packet to a new unknown destination address whose topological location in the network is unknown. An ethernet switch observes the flooding of packet to learn the topological location of an address. A switch identifies the port at which the packet from source S arrives. Then it uses this port for the packets whose destination address is source S. Thus, an ethernet network dynamically discovers the topological locations of interface addresses and dynamically builds packet forwarding tables accordingly. This mechanism is called address learning. Existing ethernet standards use a *"Spanning Tree Protocol"* which computes a cycle-free active forwarding topology to support the flooding of packets for new destination and address learning. In case of link failure, cycles in the physical topology provides a redundant path. It is necessary that the active forwarding topology to be cycle free because ethernet packets may cause congestion due to the lack of time-to-live field. When a link or switch failure occurs, it disturbs the active forwarding topology, the network leads to a packet loss. The active forwarding topology is recomputed to stop the packet loss.

The dependability of ethernet relies on the ability of the spanning tree protocol to quickly recomputed cycle-free topology upon a partial network failure. In the present day, the rapid spanning tree protocol, RSTP [1] is the dominant spanning tree protocol. But unfortunately, RSTP may exhibit the count-to-infinity problem. The spanning tree topology is continuously reconfigured during the count to infinity and it forms the temporary forwarding loop. The ports in the network can oscillate between forwarding and blocking data packets. Thus, many data packets may be dropped. The forwarding loop lasts until the count to infinity lasts. In the present work, count-to-infinity problem in RSTP is examined and an effective solution called RSTP with epochs is presented. We demonstrate the exact conditions under which the count to infinity problem occurs in RSTP and study the problem in detail with an example and identify the effects. This solution improves the convergence time of the spanning tree computation upon failure.

In this work we perform an in-depth analysis of the count-to-infinity problem and provide an effective solution. In section-II, a brief introduction of Spanning trees & BPDU's is given. Section-III discusses in detail about count-to-Infinity problem. Section-IV provides us the solution for reducing count-to-infinity called RSTP with Epochs. In Section-V, we evaluate the protocol. Section-VI describes the effects of count-to-Infinity on port saturation and also the results are evaluated. Conclusions of this work are given in section-VII.

2. Rapid Spanning Tree Protocol (RSTP)

The rapid spanning tree protocol (RSTP) was introduced in the IEEE 802.1W standard and later revised in the IEEE 802.1 D standards. RSTP was derived from Spanning Tree protocol (STP) and it is designed to overcome STP's long convergence time.

2.1. The Spanning Tree

RSTP computes a unique spanning tree over the network of bridges and connecting links. Each bridge must have a unique ID. The spanning tree is rooted at the bridge with the lowest ID. There exists only one possible path between any

nodes and it is of minimum cost. A bridge port is one that connects a link to a bridge. It has two main attributes, a 'port' and its 'state'. The port describes its role in the spanning tree. The three roles of a port are:

1) A root port connects a bridge to its parent in the spanning tree.
2) A designated port connects a bridge to one or more children in the spanning tree.
3) An alternate port connects a bridge to a redundant link that provides a backup path to the root.

A port's state describes whether the port forwards or blocks the data.

2.2. Bridge Protocol Data Units (BPDUs)

BPDUS are used to exchange topology information between bridges. Each bridge constructs its BPDUs based on the latest topology information that it has received from its parent bridge. In the absence of any new information, bridges send a BPDU every 'Hello time as a heartbeat. This heartbeat BPDUS are called Hello messages. A bridge must compare the BPDUs that it receives to use the best of these BPDUs. According to the IEEE 802.1 D standard, BPDU *m1* is better than BPDU *m2* if:

1) *m1* is announcing a root with a lower bridge ID than that of *m2*
2) Both BPDUS are announcing the same root out but m1 is announcing a lower cost to reach the root.
3) Both BPDUS are announcing the same root and cost, but *m1* was last transmitted through a bridge with a lower ID than the bridge that last transmitted *m2*.
4) Both BPDUs are announcing the same root and cost, both BPDUS were last transmitted through the same bridge, but *m1* was transmitted from a port with a lower ID than the port that last transmitted *m2*.

2.3. Building and maintaining the spanning tree

The Spanning Tree Algorithm (STA) uses the information in the BPDUs to select the root bridge which is having the lowest bridge ID and set the port roles on each bridge.

First, select the root bridge. Second, a tree of shortest paths from the root to every bridge is constructed. If a bridge fails, a new one is computed. Third, disable all other root paths. The port that has received the worst information than they are sending will become designated ports. The port that has received the best information, among all information received by all bridge ports, for a path to the root becomes the root port. A port becomes an alternate port and receives better information than it is sending.

If a root or alternate port has not received a BPDU in 3 times the '*Hello time*', the STA (Spanning Tree Algorithm) concludes that the path to the root through this port has failed and removes the information associated with this port. If a bridge detects failure at its root port, it chooses alternate port as root port.

2.4. Topology change

A topology change can result in the reconfiguration of the spanning tree. The STA implements this by making a bridge send a topology change (TC) message whenever it detects a topology change event. The bridge sends such messages on all of its ports participating in the active topology. A bridge receiving a TC message forwards this message on all of its ports participating in the active topology. The bridge receives a TC message on one of its ports; it flushes the forwarding table entries at all of its other port. Here we consider all the topologies while simulating the results.

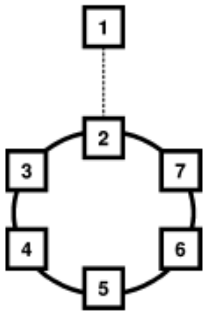

Fig.1. Simple topology vulnerable to count to infinity

2.5. Count to infinity problem in RSTP

A count to infinity can occur in RSTP when there is a cycle in the physical topology and this cycle loses connectivity to the root bridge due to a network failure. Fig.1 shows a simple topology vulnerable to count to infinity. A link failure between the bridge 1 and 2 can result in a count to infinity. This problem arises when the bridges cache topology information from the port at their alternate ports, and use the information indiscriminately in the future, if the root port loses connectivity to the root bridge. This topology information may be fresh or stale. Then the bridge may spread this stale information to other bridges resulting in a count to infinity. Here we present an example which illustrates count to infinity in RSTP which uses the following rules:

1) If a bridge can no longer reach the root bridge via its root port and does not have an alternate port, it declares itself to the root.
2) A bridge sends out a BPDU immediately after the topology information it is announcing has changed.
3) A designated port becomes the root port if it receives a better BPDO than what the bridge has received before. That is, this BPDU announces a better path to the root than via the current root port.
4) When a bridge loses connectivity to the root bridge via its root port and it has one or more alternate ports, it

adopts the alternate port with the lowest cost path to the root as its new root port.

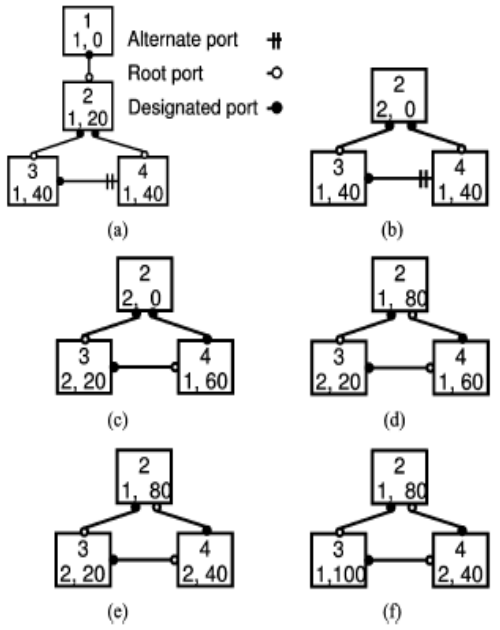

Fig.2. An example of count to infinity

(a) Time t1 (b) Time t2 (c) Time t3 (d) Time t4 (e) Time t5 (f) Time t6

Figure-2 shows a network in which each box represents a bridge. The upper number is the bridge ID and the lower two numbers represent the root bridge and the cost to the root. The cost is arbitrarily set to 20.

Figure 2 (a) shows the stable active topology at time t1. Figure 2 (b) shows the network at time t2 when the link between bridge 1 and 2 dies. Bridge 2 declares itself to be the root since it has no alternate port (rule1). Bridge 2 announces to bridge 3 and 4 that it is the root (rule 2). At time t3 bridge3 makes bridge 2 its root as it does not have any alternate port. But bridge 4 has an alternate port forming a path to bridge1. Bridge 4 incorrectly uses this alternate port as its new root port. It makes bridge 3 its parent on the path to the now unavailable bridge 1 (rule 4). Bridge 4 doesn't know that this formed topology information at the alternate port is stale. At time t4, bridge 4 announces to bridge that it has a path to bridge1, spreading the stale topology information and initiating a count to infinity. Bridge 2 makes bridge 4 its parent and updates the cost to bridge1. At time t5 bridge 3 sends a BPDU to bridge 4 saying that bridge 2 is the root. Since bridge 3 is parent for bridge 4, bridge 4 accepts this information and sets its cost to bridge 2 as 40. At time t6 bridge 2 sends a BPDU to bridge 3 saying that it has path to bridge1. Bridge 3 makes bridge 2 its parent, updating its cost to bridge 1 continues to go around the cycle in a count to infinity until either it reaches its maximum age.

Whenever a network is partitioned, if the partition does not contain the root bridge as a cycle, there exists a race condition that can result in the count-to-infinity behavior.

1) Claim 1: If a network is partitioned, the partition without the previous root bridge must contain a bridge that has no alternate port.

2) Claim 2: If a network is partitioned, and the partition without the previous root bridge contains a cycle, a race condition exists that may lead to count-to-infinity.

Count-to-infinity may even occur without a network partition. This new topology information will go around the loop until it reaches an alternate port caching stale, but better information. Again this stale information will chose the new information around the loop in a count to infinity. This will keep going until the stale topology information reaches its massage.

3. Count to Infinity Induced Forwarding Loops

The reasons for the formation of a count to infinity induced forwarding loop are.

1. Count to infinity occurs around a physical network cycle.

2. During count to infinity, the fresh topology information stalls at a bridge because the bridge port has reached its TX hold count and subsequently the stale information is received at the bridge. As a result, the fresh information is eliminated from the network.

3. The sync. operation that would have prevented a forwarding loop is not performed by a bridge because of a race condition allowing the forwarding loop to be formed.

3.1. Duration of count to infinity

A count to infinity must end when the stale information is discarded due to reaching the maximum age. Thus the stale information can cross at most maximum age hops. Topology information in a BPDU can reside in memory at a bridge for at most (3x Hello time) unless it gets refreshed by a new incoming BPDU. Therefore the theoretical upper bound for stale information to stay in the network is (3 x Hello time x Max Age).

Figure 3 shows the convergence times measured. For every number of links the experiment is repeated 10 times and report the measured conveyance times under the count to infinity. In our experiments we use a simulator [2] that is based on the simulator by Myers et al. [3]. Adding more redundant links dramatically increases the convergence time. This is because adding more redundant links results in more alternate ports per bridge.

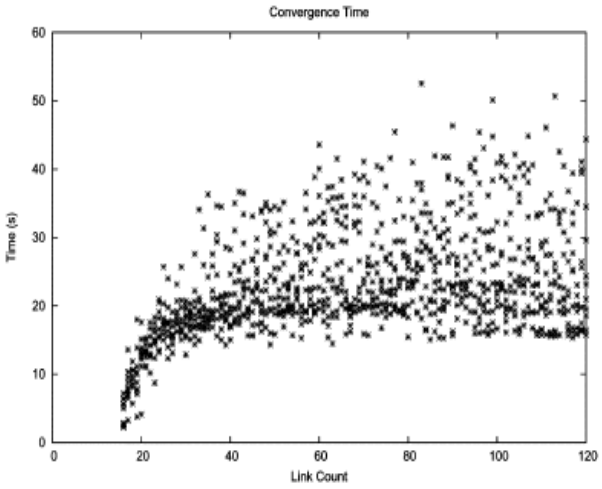

Fig.3. Convergence time in a network of 16 bridges after failure of root

4. RSTP with Epochs

RSTP with Epochs is an extension to RSTP. It mainly concentrates on introducing sequence numbers in the BPDUs. The root bridge adds a sequence number to each BPDU it generates. Then the other bridges generate and transmit their own BPDUs based on the latest root's BPDU and including the root's latest sequence number. The purpose of these sequence numbers is to identify stale BPDUs.

An Epoch is an interval starting when the true root bridge achieves root status and ending when the true root bridge contests for root status. Another bridge will contend for root status because it did not hear from the previous root or because it finds its bridge ID to be lower than that of the previous root. A bridge may find it has a lower bridge ID than the root because it has just joined the network and its bridge ID is lower than the current root's bridge ID, making it eligible to be the new root. If the previous root has retired and contending bridge is eligible to be the root, the new root will use a sequence number higher than the highest sequence number it received from the retired root.

If the old root is reachable and eligible to be the root, it will use sequence number higher than the contending bridge sequence numbers to re-take the network. Each bridge has a local representation of an epoch with an interval of sequence numbers it heard from the same root bridge. The interval is represented by two sequence numbers, *first seqno* and *current seqno*. *First seqno* is the sequence number this bridge has heard from the current root. *Current seqno* is the current or latest sequence number the bridge has heard from the root. Based on these two values the bridges will work. Each bridge records two values, *first seqno* and *current seqno*., that it has received from the current root bridge. These two sequence numbers represents the current epoch.

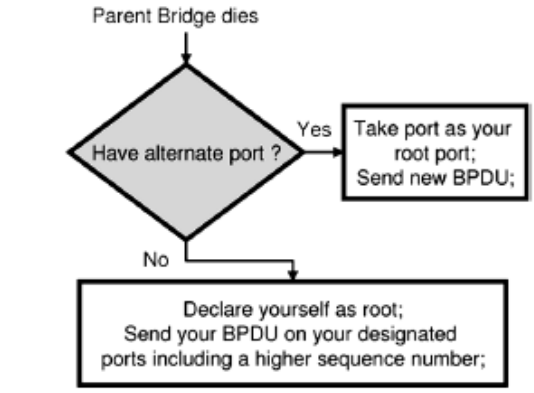

Fig.4. Handling the death of the parent bridge

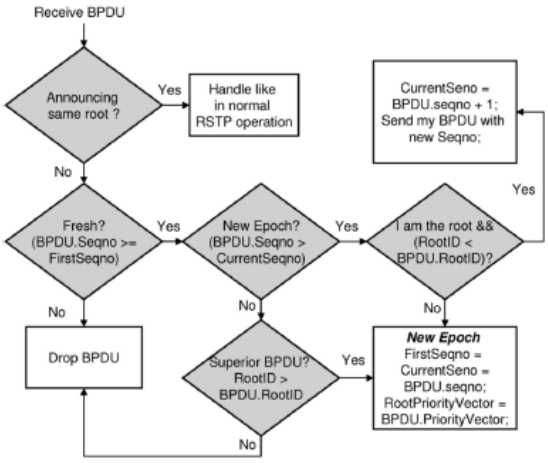

Fig.5. Handling the reception of BPDU in the RSTP with epochs

A BPDU with a sequence number less than the recorded *first seqno* must be a stale BPDU belonging to an earlier epoch. Figure 4 shows the flow chart to handle the death of the parent bridge. When a bridge detects disconnection from it parent, it goes for any alternate ports if available.It chooses one of these alternate ports as its new root port. If the bridge does not have any alternate ports, it declares itself as the new root and starts broad casting its own BPDUs that have a sequence number larger than that the last sequence number that it received from the old root.

Figure 5 shows the handling of the receipt of a BPDU in RSTP with Epochs. Bridges discard the sequence numbers when comparing BPDUS declaring the same root. If a BPDU arrives declaring a different root than the one perceived root then it signals the beginning of a new epoch. The new epoch has a different root declared by the received BPDU. The first and last sequence numbers are set to the sequence number reported by the BPDU is larger than or equal to the first recorded sequence number but smaller than or equal to the largest recorded sequence number of the current root, the bridge with the lowest ID, among the ones declared by the BPDU and the current root and it is the one accepted by the bridge as the current root.

If a bridge receives a BPDU declaring another bridge with an inferior bridge ID to its own as the root, the bridge starts sending BPDUs declaring itself as the root. These BPDUs are given a sequence number that is larger than that received from the bridge with the inferior ID. When one of these BPDUs reaches the old root bridge with the inferior ID, it will stop declaring itself as the root. Fig 6(b) shows the convergence times measure. RSTP with epochs can convergence in at most 400 microseconds in these experiments, but RSTP takes second to converge even under this simple network. In the third set of experiments we use simple "Ring" topologies where the bridges form a simple cycle. Again we kill the Root Bridge and measure the convergence time for both protocols. Figure 6(c) shows the convergence times measured. Here RSTP with Epochs takes roughly twice the amount of time to converge compared to RSTP.

6. Effect of Count to Infinity on Port Saturation

A port is said to be saturated if it has reached its TX hold count limit but still has more BPDUs to transport. We present a time sequence of the number of saturated ports in the whole network in the three experiment scenarios.

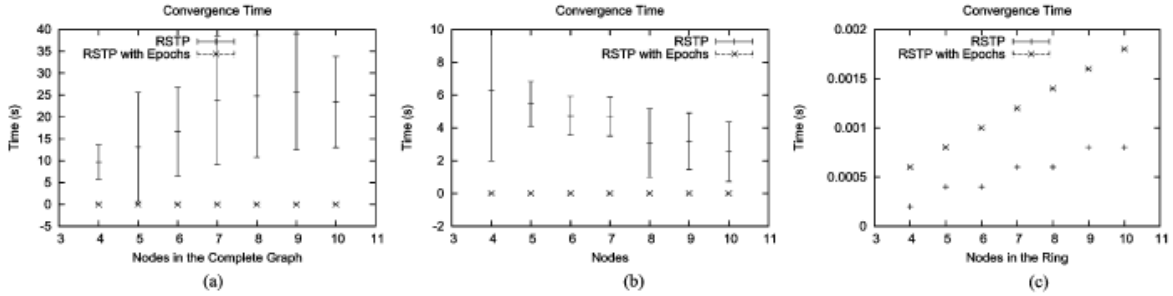

Fig.6. Convergence Time in a network of 4 to 10 bridges (a) Complete Graph topologies (b) Loop topologies (c) Nodes in the ring

5. Evaluating RSTP with Epochs

In this section, we compare the convergence times of RSTP and RSTP with epochs in the event of failure in three families of topologies. For each family of topologies, the number of bridges in the network varies and the corresponding convergence time is measured.

In the first experiment we take a set of complete graphs varying the number of bridges in the network. In each run we kill the root bridge and measure the time. It takes for the network to converge under both protocols.

Fig 6(a) shows the convergence times measured. Here the vertical bar represents the range of values measured for each network size. In the graph, the highest convergence time observed for RSTP with epochs is only 100 micro seconds. RSTP with epochs does not suffer from the count-to-infinity problem on the other hand; RSTP takes a much longer time to converge.

In the second set of experiments we use simpler loop topologies. For example, a network with 10 bridges means the loop has 9 bridges and the loop is connected to the root bridge that does not lie on the loop. Again, we kill the Root Bridge and measure the convergence time for both protocols.

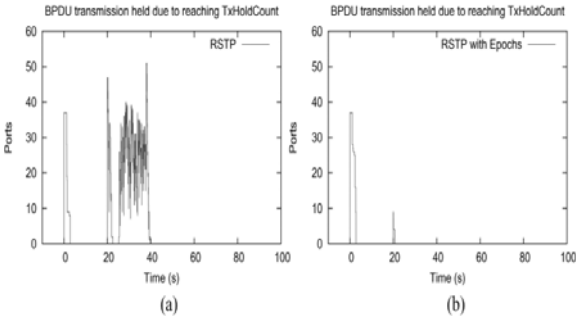

Fig.7 Time sequence of number of ports that reached their TxHoldCount limit while still having more BPDUs waiting for transmission. This experiment is for a 10 bridge fully connected graph topology where the root bridge dies at time 20 (a) RSTP (b) RSTP with Epochs. In the first experiment a complete graph of 10 nodes.

Figure 7 (a) shows the spike in the number of saturated ports at startup. Due to spike in the transmitted BPDUs at startup by both protocols, starting from time 20 when the root port dies, we find a long period of time that is close to 20 seconds in RSTP. This is due to count to infinity problem where BPDUs spin around the loop causing the ports to quickly reach their TX hold count limit. But RSTP with Epochs does not suffer from the count-to-infinity problem.

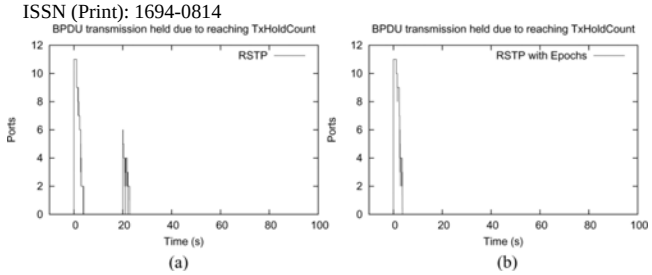

Fig.8 Time sequence of number of ports that reached their TxHoldCount limit while they still have more BPDUs waiting for transmission. This experiment is for a 10 bridge "loop" topology. The root bridge dies at time 20. (a) RSTP. (b) RSTP with Epochs.

Similarly in the second experiment, we observe from figure 8 (a) the spike in the number of satured ports at startup. We also observe in RSTP a period after the failure of the root bridge where there are several saturated ports. Again this is due to the count-to-infinity problem.

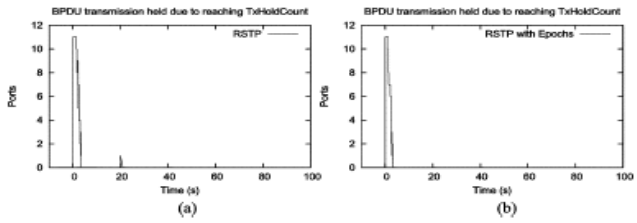

Fig. 9 Time sequence of number of ports reaching their TxHoldCount limit while still having more BPDUs waiting for transmission. This experiment is for a 10 bridge ring topology where a link connecting the root bridge to a neighbor dies at time 20. (a) RSTP. (b) RSTP with Epochs.

In the third experiment, failure of the root cuts the loop so there is no count to infinity. Thus for both protocols no ports get saturated after the failure that you can see in fig.9

7. Conclusions

In studying RSTP under partial network failure, it can exhibit a count-to-infinity problem. In experiments, we observe that in some scenarios the count to infinity can extend the convergence time to reach 50 seconds. During the count to infinity, bridges transmit a lot more BPDUs than during its normal operation. In this work, we identify the exact conditions under which the count-to-infinity problem arises. We propose a simple and effective solution called RSTP with epochs that eliminate the count to infinity problem. This solution also enhances the convergence time.

References

1) "Spanning tree protocol problems and related design consideration," Cisco systems, Inc. (online). www.cisco.can/ward/public/473/16.htm/

2) K. Elmeleegy, "RSTP with Epochs simulator", 2007 (online) http://www.cs.rice.edu/kdiaa/ethernet

3) K. Elmelegy, A. L. Cox, and T.S.E.Ng, "On count-to-infinity induced forwarding loops in Ethernet networks," in Proc. IEEE Infocom 2006, Barcelona, Spain, Apr. 2006.

4) K. Elmelegy, A.L.Cax, and T.S.E.Ng, " Etherfuse: An Ethernet watch dog," in Proc. ACM SIG Communications, Kyoto, Japan, Aug.2007, pp. 253-264.

5) Khaled Elmeleegy, Alan L. Cox, and T. S. Eugene Ng. "Understanding and Mitigating the Effects of Count to Infinity in Ethernet Networks". *To appear in IEEE/ACM Transactions on Networking.*

6) R. Garcia, J. Duato and F.Silla "LSOM:A link state protocol over MAC addresses for metropolitan backbones using optical Ethernet switches" in *proc.* 2nd *IEEE Int .Symp. Network Computing and Applications* (NCA'03), pp.315-321, Apr.2003.

7) J. J. Gracia-lunes-Aceves, "Loop-free routing using diffusing computations," IEEE/ACM Trans. on Networking, vol.1, No.1, pp.130-141, Feb.1993

8) J. M. Jafee and F. H. Moss, "A responsive distributed routing algorithm in computer networks," *IEEE Trans. on Communications.*, vol. 30, No.7, pp.1758-1767, July 1982

9)R.Pelman "RBridges:Transparent Routing,"in proc.IEEE IN-FOCOM,2004,vol.2,pp.234-244

10)S.Ray,R.A.Guerin and R.Sofia "Distributed path computation with out Transient loops" ," *IEEE Trans. on Communications.*, vol. 30, No.7, July

Mr. D. Ganesh received his B.Tech degree in Information Technology from JNT University, Hyderabad in 2006 and M.Tech degree in Computer Science and Engineering from Acharya Nagarjuna University in 2010.During the period 2006-07 he worked as Assistant Professor in Information Technology department at AITS, Rajampet, India. Since 2007, he is working as Assistant Professor in IT Department at Sree Vidyanikethan Engineering College, Tirupati, India. He has Published 4 papers in national and International conferences. His current research interests are computer networks, Object oriented design and unified modeling. He is a member of ISTE, CSI

Dr. Rama Prasad V Vaddella was born in Tirupati, India in 1962. He received the M.Sc (Tech.) degree in Electronic Instrumentation from Sri Venkateswara University, Tirupati in 1986 and M.E degree in Information Systems from BITS, Pilani, India in 1991. During the period 1989-1992 he worked as Assistant Lecturer in Computer Science at BITS, Pilani. From 1992 to 1995, he worked as Lecturer in Computer Science and Engineering and as Associate Professor from 1995 to 1998 at RVR & JC College of Engineering, Guntur, India. Since 1998, he is working as Professor and Head of Information Technology department at Sree Vidyanikethan Engineering College, Tirupati, India. He was awarded the Ph.D degree in Computer Science by J.N.T. Univeristy, Hyderabad, during 2007 for his thesis in Fractal Image Compression. He is also holding the additional position of Chairman, Board of Studies in Computer Science, and Vice-Principal at Sree Vidyanikethan Engineering College (Autonomous under JNT University, Anantapur, India). He has also worked as a short time Research Assistant at Indian Institute of Science, Bangalore during the year 1986. He has published about 10 papers in national and international conferences and 05 papers in International journals, edited books, and refereed conferences. He is also a member of the editorial review board for 05 International journals in Computer Science and Information Technology. His current research interests include computer graphics, image processing, computer networks, computer architecture and neural networks. He is a member of IEEE, ISTE and CSI