IJCSN International Journal of Computer Science and Network, Volume 5, Issue 5, October 2016
ISSN (Online) : 2277-5420 www.IJCSN.org
Impact Factor: 1.02

Exploring Multiple Paths using Link Utilization in Computer Networks

[1] **Shalini Aggarwal,** [2] **Shuchita Upadhyaya**

[1] Teacher Fellow, Department of Computer Science & Applications, Kurukshetra University
Kurukshetra, Haryana - 136119, India

[2] Professor, Department of Computer Science & Applications, Kurukshetra University
Kurukshetra, Haryana - 136119, India

Abstract - Though the speed of the Internet is increasing day-by-day, yet there is need of much higher speed. A lot of research is going on to receive any information from Internet just on a click. One way to have more efficient and robust Internet is to identify multiple paths for routing so that routers could flexibly divide traffic over these paths. Often having one or two more paths is generally enough for significant gains in terms of security, performance, and reliability of a network. However, one of the first barriers in multipath routing is to explore the multiple paths over which traffic can be diverted. The principle objective of this paper is to explore the multiple paths using link utilization factor. For this purpose a multi-path routing algorithm has been proposed that identifies multiple loop-free paths from single source to single destination using local information. The link utilizations factor used in this algorithm takes into account various metrics such as load, capacity and delay etc. The proposed technique identifies multiple paths which may not be over utilized.

Keywords - *Link Utilization, Load, Capacity, Multipath Routing.*

1. Introduction

Most of the currently used routing protocols select only a single path for the traffic between each source-destination pair. Multipath routing finds up multiple routes between source and destination. Internet-wide multipath routing would offer a number of advantages, such as:

- Different applications have different needs. If more than one path exists, the audio or video streaming and online-gaming traffic may use a low-delay path so that buffering is minimum, while file-sharing traffic may use a path which may yield higher throughput. In addition, if an application requires higher bandwidth, it can access more bandwidth by using more than one path simultaneously.
- Improving end-to-end reliability: If multiple paths exist, traffic can switch quickly to an alternate path when a link or router fails.
- Avoiding congested paths: If multiple paths are available, traffic can move to an alternate path when it observes congestion within a path. The traffic load can be distributed in such a manner so that no path is congested.
- Quality of Service: Another important benefit is QOS achieved when using multipath routing.

The above-said advantages of multipath routing are also supported by various studies. A study showed that, although Internet traffic traverses a single path, 30% to 80% of the time, an alternate path with lower loss or smaller delay exists.

In this paper, a new technique has been proposed that explores loop-free multiple paths so that flexible division of traffic over multiple paths can be done. The multiple paths are obtained using the link utilization factor. The link utilization factor is based on load, capacity and delay. The multiple paths found are stored at destination node. The proposed work has been implemented in MATLAB and manually verified.

The paper has been divided into 7 sections. Section 2 presents some of the related literature. Section 3 defines the problem. Section 4 presents proposed work along with the algorithm. Section 5 presents the experimental setup and results for example network. This section also presents the proof of correctness for example network. Section 6 interprets the results while section 7 concludes the paper and presents the future work.

2. Related Work

For Multi-path Identification, various algorithms have been suggested in literature as following:-

Equal cost multipath (ECMP)[1] routing has been proposed for distributing routing packets among multiple paths of equal costs. However, (i) ECMP is not guaranteed to determine a multipath route for each source-destination pair; (ii) The characteristics of the multipath route are not taken into account; (iii) Packets are forwarded in equal proportion, on a packet-by-packet basis, over the paths in the multipath route, without considering network congestion.

MPATH[2] is one of the first known routing algorithms based on distance vectors that (i) provides multiple loop free paths of variable cost to each destination in steady and dynamic environment, and (ii) uses a synchronization mechanism that is based on hop by hop routing, and therefore is more scalable as compared to other routing algorithms based on same technique.

A solution of unconstrained problems using shortest path technique has been presented in [3]. To find the K-shortest path, various shortest path algorithms such as Dijkstra & Bellman-Ford-Moore algorithms are used in the modified forms. The first type of algorithm ensures that not only the final paths but also the subpaths are shortest. The second type of algorithm finds the set of K-shortest paths using graph and tree theory.

[4] Presents the algorithm that is not necessarily loop less. The algorithm is based on breadth-first search and priority queues. The scheme presented in this algorithm is to create a binary heap for each vertex, recording the edges that are not part of the shortest path tree and that can be reached from that vertex by shortest-path-tree edges.

MPDA[5] is an algorithm which is based on link state. MPDA is based on PDA (Partial Dissemination Algorithm) to find shortest routes from source to destination. This algorithm is a loop free algorithm which is an improvement over PDA. In MPDA, each Link State Update message sent by a node is acknowledged by all its neighbors before the node sends the next LSU. The inter-neighbor synchronization used in MPDA spans only a single hop, unlike the synchronization in diffusing computations which potentially spans the whole network and therefore may result in congestion and can prove to be more time-taking.

Yen's algorithm[6] finds loop less paths that have shortest lengths from one node to another node in a network. The algorithm takes every node in the shortest path except the terminating node and calculates another shortest (spur) from each selected node to terminating node. For each such node, the path from the start node to the current node is the root path. Two conditions are placed on the spur path. First, it must not pass through any node on the root path (loop less) and secondly it must not branch from the current node on any edge used by previously found shortest paths with the same root (disjoint). When a new spur path is found it is added to the root path for that node. That path is next candidate shortest path. All such paths are stored. The same process is repeated calculating spur paths from each node in each new k shortest paths until the required no of K paths have been found. The major significance of this algorithm is that it upper bound varies linearly with the value of k.

An algorithm to find the k-shortest loop less paths has been described in [7]. The basis of this algorithm is replacement paths. Although, the replacement paths subroutine is found to fail for some directed graphs, yet the failure can be easily detected. By using this replacement path technique, the k-shortest paths have been found in this work.

[8] Presents distributed QoS multi-path routing algorithm (DQM) that finds multiple disjoint paths in a distributed way from one source to one destination. These multiple paths satisfy the given QoS requirements. The QoS parameters that have been considered are residual bandwidth and delay.

3. Problem Formulation

One of the major concerns in a network is how to transmit the data from source to destination at a faster rate. For this purpose one of the techniques that may be followed is multipath routing. In this paper, a novel technique to find multiple loop-free paths from a given source to destination has been presented. This paper is not finding all the possible paths from source to destination; rather it is confined to find paths which are better as compared to other path depending on the link utilization. Link utilization itself is based on various known metrics of the network such as bandwidth, delay and load.

4. Proposed Work

In this work, various loop free paths are explored depending on the various known metrics for the network. The multiple paths are selected according to the link utilization factor, which is derived from the concept of choke packets in controlling congestion [9].

$u_{new} = a*u_{old}+(1-a)*f$……………….(1)

Where $\mathbf{u_{new}}$ is new calculated value of link utilization,
The value of **'a'** indicates how fast a node forgets recent history,

$\mathbf{u_{old}}$ is the previous value of link utilization,

f indicates instantaneous line utilization.

Each node monitors the percent utilization of each of its output lines associated with each line.'u' reflects the utilization of that line. To maintain a good estimate of 'u', a sample of the instantaneous line utilization f is taken to be 0 or 1.

a = capacity / load,

Where capacity indicates the capacity of the link, and load indicates the current load on the link.

Here value of 'a' represents how fast a node forgets its current history. It is calculated on the basis of current values of load and capacity of that link and therefore it helps in forgetting the historic values of the node.

The paths will be identified according to proposed strategy. The advantage will be that instead of one optimal (shortest) path more than one optimal paths are found so that traffic can be distributed over multiple paths.

4.1 Design of the Proposed Algorithm

A network is modeled as a set V of nodes that are interconnected by a set E of full-duplex, communications links.

Information Stored At Each Node: At each node, the following information is required to be stored.

The local information to be stored at each node i
$\mathbf{n_{i,j}}$ -List of adjacent nodes of i , to which data packets for destination may be forwarded.
$\mathbf{c_{i,j}}$ - Capacity of outgoing links from i.
$\mathbf{l_{i,j}}$ - Load of outgoing links from i.
$\mathbf{u}_{i,j}$ – Link utilization of outgoing links from i.
Here j indicates all the neighbor nodes of node i.

Routing Table: Since multiple paths are being calculated, the routing table stores multiple values relative to each path. The routing table contains the following information related to identified path at each intermediate node and at destination as well-

Cid- Connection id is the unique identification number assigned by the source to represent routing request.
PRED (i,k)- It stores predecessor for node i in the kth path.
PT (i,k) - It stores the intermediate nodes of kth path from source to that node i.
$\mathbf{u_k(i, j)}$ - link utilization value of link from node i to node j in kth the path.
Here k varies from 1 to total number of paths identified at that node.

Message Format: The information in the EXPLORER packet includes the connection identification, Source, Destination, Sender of message, the Link Utilization, Path list and the number of hops in the path.

The format of the EXPLORER message has been shown in the Table 1.

Table 1 EXPLORER Message Format:

SA	DA	PRED	Cid	u	PT

The fields of Probe message represent the following:
SA – Source Address
DA – Destination Address
PRED – Predecessor Address (Sender of the Packet)
Cid – Connection Identification
u – Link utilization
PT – Path list traversed so-far

In the proposed algorithm, the various actions performed at the source, intermediate and destinations nodes are explained below:

Source Node: Source node will send the explorer message to its entire outgoing links depending upon the value of line utilization u. The source node will be added to the path list.

Intermediate Node: Each neighbor of the intermediate node will be again checked for the u_{thres} value. Intermediate node will forward the explorer message to all outgoing links that are satisfying the line utilization threshold value and do not exist in the current path list so that loop can be avoided. Each node will maintain a routing table. It will calculate the corresponding value of the line utilization of all adjacent links by looking into its local information table. If line utilization value is less than the threshold value u_{thres}, then the new value of u is calculated according to u_{old}. Accordingly, it will update the message for all its qualifying neighbours by adding name of intermediate node in the path list.

At Destination node: When the first explorer message of a particular connection ID is received by the destination node, it stores the Cid, sender, the source node and the complete path in the routing table and waits for a pre-specified time period for more paths.

4.2 Proposed Algorithm: The proposed algorithm finds multiple loop free paths

```
(i)    Initialize variables. //Source, Destination, $u_{Thres}$,
       Load and Capacity, f.
(ii)   Initialize u with zero for all nodes.
(iii)  Path = Source
       //At the source node
(iv)   For every node i which is a neighbor of Source do
               f = 1
               a=Capacity(source,i) /Load(Source,i)
               //Compute  value of u
               If u < 0 then u = 0
               If(u < $u_{thres}$)
                 Send Explorer message to i.
                 Add node i to the path list
              Endif
       End for
       //At the intermediate node
(v)    For every node j which is a neighbor of an
       intermediate node i and does not exist in the path
       list do
                 f = 1
                 a = Capacity (i, j) / Load (i, j)
                 Compute u
                 If u < 0 then u = 0
                 If (u < $u_{thres}$)
                     Send Explorer message to j.
                     Add node j to the path list
                 Endif
       End for
       //At the destination node
(ix)   Destination node stores all the explorer messages
       information in its routing table
(x)    End.
```

5. Experimental Setup and Results

The proposed algorithm is implemented using MATLAB for the network shown in fig 1. The input given to the program is load and capacity (bandwidth). The network of five nodes is considered for this experiment.

The input is given in three different variations
(i) Small load. (ii) High load and (iii) Random load.
The capacity is assumed to be 10 in every case.
The threshold value for the line utilization u is assumed to be 0.7.

Case (i): Load is shown on each edge in fig. 1. The load on each link is assumed to be smaller than capacity.

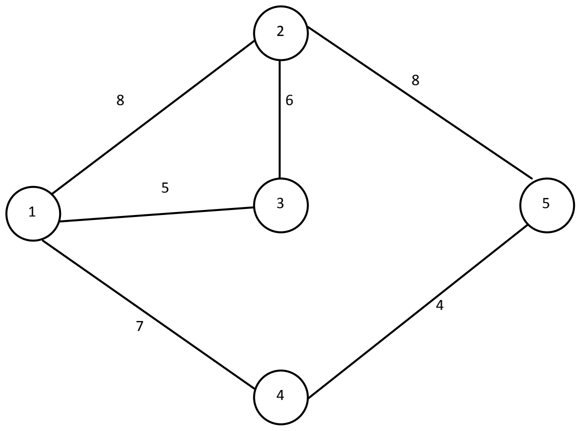
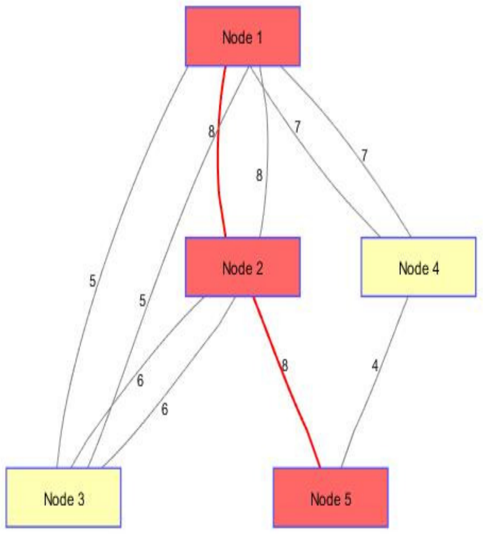
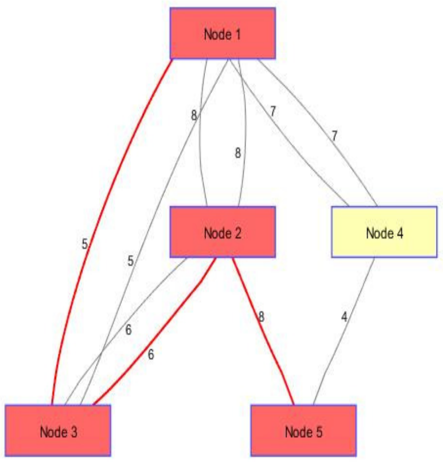


Fig. 1: Network with Load shown on the links

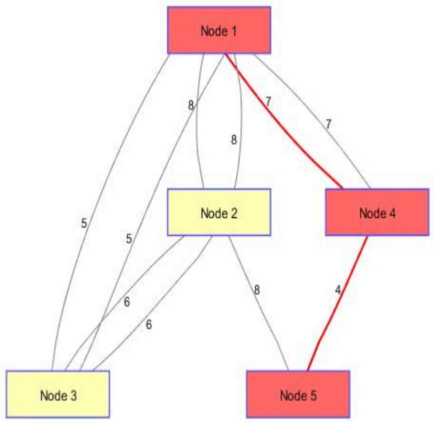
On the basis of these inputs, the resultant paths identified are shown in Fig 2:



Path1 (1-2-5)



Path2(1-3-2-5)



Path3 (1-4-5)

Fig. 2: Paths found in Network shown in fig. 1

Case(ii) : If load is higher than capacity

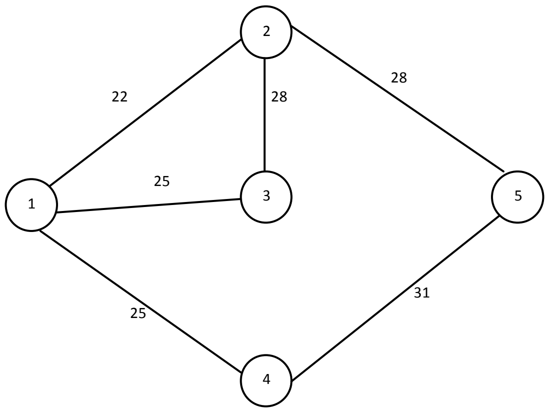


Fig. 3: Network with Load shown on the links

Path identified in this case (only 1 path is identified):
Path1: 1 2 5

Case (iii): If the load is taken randomly as shown in fig. 4:

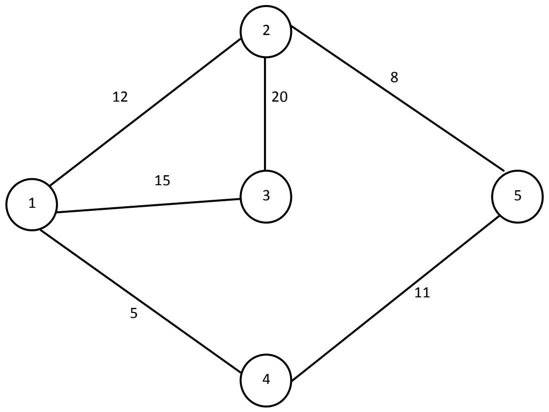
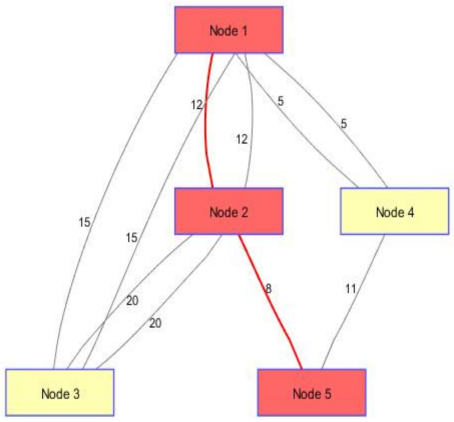
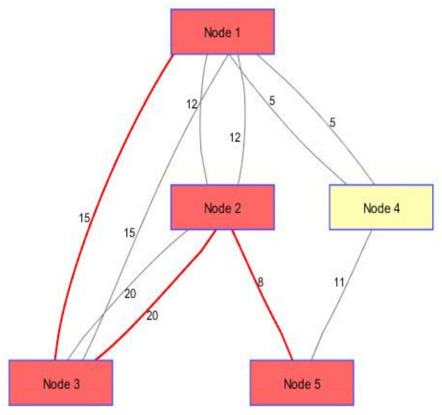


Fig. 4: Network with edges depicting assumed link Load

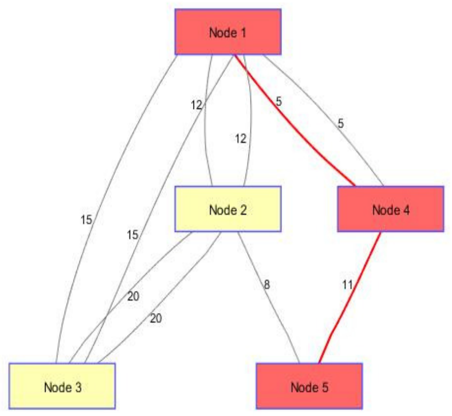
The graphs for the identified paths are shown below in Fig.5.



Path1(1-2-5)



Path2(1-3-2-5)



Path2(1-3-2-5)

Fig. 5: Paths found in Network shown in fig. 4

5.1 Proof of Correctness

For proving the correctness of proposed algorithm, the loop invariants need to be verified. Here the verification for case (iii) is presented. Similarly case (i) and case (ii) can be proved.

Verification for case (iii): In the proposed algorithm two loop invariants exist. First loop invariant is at step (iv) while the other is existing at step (v).
Three properties needed to be proved for proving the correctness of this loop.

Initialization: At the beginning of the loop, the path list consists of only source, which should be the case and shows that the loop invariant holds prior to the first iteration of the loop.

Maintenance: In this case the value of u is calculated for each neighbor of source node and on the basis of whether $u < u_{Thres}$, explorer message is transmitted and the neighbor is added to the path list. In case (iii) of section 5, consider the network shown in fig. 3,
The link utilization values for various links from the source are calculated as below:

Link 1-2: Since f and u are initialized to be zero, therefore u(1) will be zero according to the formula. u(2) will be calculated as follows
f = 1, a = 10/12 = 0.833, Value of u stored at node (2) = 0.833 * 0 + (1 - 0.833) * 1 = 0.166.

Link 1-3:
f = 1, a = 10/15 = 0.666, Value of u stored at node (3) = 0.666 * 0 + (1 - 0.666) * 1 = 0.333

Link 1-4:
f = 1, a = 10/5 = 2, Value of u stored at node (4) = 2 * 0 + (1 - 2) * 1 = -1 ≈ 0

As in all the cases the value of $u(i) < u_{thres}$, therefore explorer message will be transmitted further.

Therefore, it can be seen that second property also holds for this loop.

Termination: The loop is terminated when all the neighbors of source node have been checked for link utilization. Hence third property also hold good.

Hence the loop invariant is correct.

Now second loop invariant will be proved on the basis of these three properties.

Initialization: At the beginning of the loop, the path list consists of source and its immediate neighbours, which should be the case prior to this loop and it shows that the loop invariant holds before the first iteration of the loop.

Maintenance: In this case the value of u is calculated for each neighbor of intermediate node and on the basis of whether $u < u_{Thres}$, explorer message is transmitted and the neighbor is added to the path list. Following calculations shows the results:

Link 2-3:
f = 1, a = 10/20 = 0.5, Value of u stored at node (3) = 0.5 * 0.333 + (1 - 0.5) * 1 = 0.666.

Link 2-5:
f = 1, a = 10/8 = 1.25, Value of u stored at node (5) = 1.25 * 0 + (1 - 1.25) * 1 = -0.25 ≈ 0 < u_{thres}

Since node 5 is destination node, first path i.e. 1-2-5 has been identified.

Now the algorithm will search path from 1-2-3 and further. But as per the algorithm no nodes can be repeated and from node 3 only links available are 3-1 and 3-2 and both node already exist in this path, therefore this path will no longer be explored and discarded.

Now the explorer message which was transmitted from source node to the node 3 will move further.

Link 3-2:
f = 1, a = 10/20 = 0.5, Value of u stored at node (2) = 0.5 * 0.166 + (1 - 0.5) * 1 = 0.583.

Link 2-5:
f = 1, a = 10/8 = 1.25,
Value of u stored at node (5) = 1.25 * (-0.25) + (1 - 1.25) * 1 = -0.562 ≈ 0 < u_{thres}

Again node 5 is destination node; second path i.e. 1-3-2-5 has been identified.

Now the explorer message which was transmitted from source node to the node 4 will move further.

Link 4-5:
f = 1, a = 10/11 = 0.909,
Value of u stored at node (5) = 0.909 * (-0.562) + (1 - 0.909) * 1 = -0.5 ≈ 0 < u_{thres}.

Again node 5 is destination node; third path i.e. 1-4-5 has been identified.

In this case the path where node already exists in the path list is discarded as stated above and only those explorer messages where the constraint on u is satisfied are followed, which proves the maintenance property as well.

Termination: The loop is terminated when all possible paths are found and destination node is reached. Hence third property also hold good.

Since both the loop invariants are proved to be correct. Hence the proposed algorithm is working correctly.

All the links has been traversed, therefore all of the paths which follow constraints in the algorithm are identified which are 1-2-5, 1-3-2-5 and 1-4-5. These results coincide with the execution of algorithm using MATLAB, therefore it can be clearly observed that algorithm and its implementation is working correctly.

Similarly case (i) and case (ii) can be verified.

6. Interpretation of the Results

On the basis of the example network, it can be observed that when the load on the network is random i.e. at some node load is greater than capacity and at some node load is smaller than capacity, then the number of paths identified are neither the maximum numbers of possible paths, nor very less number of paths. The paths which are having less utilization are identified for the future traffic. While when the load is quite high as compared to capacity on all the links, then the paths identified are very less and only the paths which are having comparatively lesser utilization are found. In the third case when the load is very low or approximately equal to the capacity, then in most of the cases all the possible paths are identified.

Also it can be seen that the paths where utilization is less are identified in every case, while the paths having more utilization are rarely found, which is an obvious requirement for routing. Overall it can be said that the proposed algorithm is able to identify better paths for future transmission of data over the network.

7. Conclusion and Future Scope

In this paper an algorithm is proposed to identify multiple paths for routing in a computer network. The algorithm which is based on link utilization is able to find multiple good paths instead of single best path, so that overall data transmission is faster. The algorithm was verified using example network and the results were according to the expectation. In future out of these multiple paths a few better paths can be found so that data is transmitted over a fewer paths rather than the entire identified paths.

References

[1] Moy, J., OSPF Version 2, RFC2328, 1998.
[2] Srinivas Vutukury, An Algorithm for Multipath Computation using Distance-Vectors with Predecessor Information
[3] Martin, R., Menth, M. & Hemmkeppler, M., Accuracy and Dynamics of Hash-Based Load Balancing
Algorithms for Multipath Internet Routing, IEEE, June, 2006
[4] Eppstein, D., Finding the k Shortest Paths, March 31, 1997
[5] Srinivas Vutukury, Multipath routing mechanisms for traffic engineering and quality of service in the

Internet, Ph.D. thesis, University of California, Santa Cruz, March 2001

[6] Yen J.Y., Finding the K-shortest loopless paths in a Network, Management Sciences, Vol.17, No.11, July 1971.

[7] Hershberger, J., Maxel, M. & Suri, S., Finding the K shortest paths: A New Algorithm and its Implementation, ACM Transactions on Algorithms, Vol. 3, No. 4, Article 45, Publication date: November 2007.

[8] Devi Gaytri and Upadhyaya Shuchita, An Approach to Distributed Multi-Path QoS Routing, Indian Journal of Science and Technology, Vol 8(20), August 2015..

[9] Andrew S. Tanenbaum and David J. Wetherall. Computer Networks, Third Edition, Prentice Hall.

Shalini Aggarwal is currently doing her Doctoral Degree in the field of Computer Networks. She has done her MCA in 2005 and M.Sc. in Computer Science (Software) in 1998 from Kurukshetra University, Kurukshetra. She is currently working as an Assistant Professor in Computer Science at Govt.(PG) College for Women, Karnal. She is having teaching experperience of more than 12 years.She has presented papers at various National and International conferences.

Shuchita Upadhyaya is currently working as a Professor in the Department of Computer Science & Applications, Kurkshetra University, Kurukshetra. She has done her Ph.D. in Computer Science & Applications from Kurkshetra University, Kurukshetra. She is having teaching & research experience of more than 28 years. She has published more than 100 papers in international and national journals.She is life member of Indian Society of Information Theory and Applications (national) and international Forum for Interdisciplinary Mathematics.She has been awarded ***Kunj Ratan,*** Award of Honour, for exemplary Achievement in the field of Academics.