In Proc. Second International Symposium on Uncertainty Modeling and Analysis, College Park, MD (April 1993), pp. 286-293.

A Fuzzy Set Delay Representation for Computer Network Routing Algorithms

Sridhar Pithani and Adarshpal S. Sethi
Dept. of Computer and Information Sciences
University of Delaware
Newark, DE 19716, USA

Abstract

Routing is an important functional aspect of a data communication network that handles data packets during their transit from source to destination. The efficiency of a distributed dynamic routing algorithm depends on how well it adapts to the traffic and topology changes in the network. The nodes in the network periodically exchange information regarding these changes for efficient routing. The information used by the algorithms to arrive at the routing paths suffers from uncertainty due to a variety of reasons. In this paper, we apply certain aspects of fuzzy sets to model the uncertainty in the delay information and use this new delay representation to improve the performance of one particular class of routing algorithms in computer networks.

1 Introduction

Traditionally, mathematical models have been used to solve many real world problems. In conventional mathematical models, it is not possible to capture all aspects of the real world phenomena. In particular, lack of certain types of information, errors in measurement while collecting data, and at times the presence of incorrect information, all give rise to uncertainty and undermine the confidence in the results obtained from the models. Proper representation of uncertainty and its incorporation into the models could be of genuine use in constructing more realistic models of real world problems.

The mathematical representation of uncertainty has received considerable attention [2, 6, 7] since the introduction of Fuzzy Set Theory by Zadeh [13]. The concept of Fuzzy Sets is based on the grouping of the elements into classes that do not have sharply defined boundaries. These classes are used to describe ambiguity and vagueness in mathematical models of empirical phenomena. A binary system is inadequate to represent these systems because certain aspects of reality always escape these models. A statement can be partially true and its truth value is treated in a relativistic sense rather than the absolute sense. This allows large flexibility in inferring information and in modeling problems that are too complex or ill-defined to be modeled otherwise. Fuzzy Set Theory provides for a mathematical representation of ambiguity and level of perception of the truth value of information.

Routing is an important functional aspect of a data communication network that handles data packets during their transit from source to destination. A routing algorithm sets up paths to connect the different nodes in the network and strives to optimize performance measures like mean packet delay or network throughput. The information used by these routing algorithms to arrive at these paths suffers from uncertainty due to a variety of reasons. In this paper, we apply certain aspects of fuzzy sets to model the uncertainty in the delay information and use this new delay representation to improve the performance of one particular class of routing algorithms in computer networks.

Routing algorithms in computer networks can be classified in different ways [1, 11]. A distributed dynamic routing algorithm is one in which each node in the network makes decisions regarding the optimal paths independently of other nodes and is adaptive to the changes in the traffic patterns and topology of the network. All nodes in the network periodically exchange information in a manner determined by the routing algorithm. This information is then used by each node to make its own routing decisions.

The efficiency of the routing algorithm depends on how well it adapts to the traffic and topology changes in the network. The information regarding these changes in network status is crucial for efficient routing of data. There is inherent uncertainty in this

information because it is at least as old as the propagation delay between nodes. The changes in the status of the network are not immediately known to all the nodes in the network. The information available at the nodes at any time is often out-of-date as there is a delay between the occurrence of an event and the arrival of this information at all the nodes. This uncertainty is higher for information about nodes that are many hops away. Traditional network routing algorithms have not attempted to deal with this uncertainty in any way and mostly behave as if there is no uncertainty in the information available at a node. Consequently, these algorithms often make poorer decisions when trying to route to distant nodes.

In this paper, we use fuzzy sets to represent this uncertainty and incorporate the uncertainty estimates into the criteria for choosing the minimum delay paths. In the next section, we briefly describe two different classes of routing algorithms used in computer networks and then explain why it is important to account for the uncertainty in the information exchanged by the nodes. In section 3, we introduce a new delay representation using fuzzy numbers and describe our proposed modified routing procedures based on this delay representation. In section 4, we present the results of some simulation studies of these algorithms.

2 Routing Algorithms in Computer Networks

A computer network may be thought of as a set of nodes connected by communication channels or links. Each channel is characterized by a transmission speed at which data can be sent on it and a propagation delay which is the delay between transmitting the data at one end and its arrival at the other. Data to be sent from a source to a destination is formatted into packets which may have a limit on their maximum size. Each packet is handled by the network independently of other packets. A routing algorithm is used to decide the path to be taken by a packet in traveling from the source to the destination.

As mentioned in the previous section, a distributed routing algorithm operates independently at each node. When a packet arrives at a node, the routing algorithm decides the next node to which it should be sent. This is equivalent to determining the outgoing link from the current node on which the packet should be transmitted. A queue is maintained of packets waiting to be transmitted on each outgoing link. The new packet is added to the end of this queue and will get transmitted when all packets in front of it have been sent. When the packet arrives at the next node, the node first checks to see if it is the destination. If it is, the packet is removed from the network; if not, the same procedure is repeated at this node.

The decision on the next node to which a packet is sent is made by looking up a table called the Route Table. The Route Table at a node contains an entry for each possible destination; thus its size is the number of nodes in the network. For each destination, the table contains the identity of the next node on the "best" known path to that destination. This is the node to which the packet is sent next. The notion of "best" path is often based on the total delay to the destination which is the sum of the propagation delays along the links traversed by the packet and the queuing delays in each of the nodes on the path. The queuing delays depend on the transmission speeds of the outgoing channels and on the other traffic that may be traveling in the same direction.

The most important part of a routing algorithm is the construction of the Routing Tables at all the nodes of the network. This is done in a distributed algorithm by periodically exchanging routing information (called an *update*) between the nodes. This makes the algorithms dynamic because the information can reflect changes in network topology due to node and link failures and recovery and changes in queuing delays because of traffic variations. Depending on the type of information exchanged, these algorithms are classified into two classes: *distance-vector* algorithms and *link-state* algorithms. In a distance-vector algorithm, each node exchanges updates only with its immediate neighbors. The node receives from each of its neighbors a distance-vector containing the length of the shortest path (distance) from that neighbor to every destination. It then uses these updates to compute its own distance-vector containing the distance from itself to every destination, and uses this vector to revise its own Routing Table. In a link-state algorithm, each node must know the entire network topology and updates are broadcast by each node to every node in the network. The update sent by a node contains the state of each of its adjacent links as well as the delays in its own outgoing queues.

An example of a distance-vector algorithm is the distributed Bellman-Ford algorithm [1, 14] that was used in the old Arpanet routing protocol. This algorithm suffers from a number of problems that include routing-table loops and counting to infinity. But some distance-vector algorithms have been recently proposed that eliminate these problems [3, 5, 8]. Ex-

amples of link-state algorithms are the distributed version of Dijkstra's shortest-path algorithm as implemented in the new ARPANET routing protocol, and the more recent IS-IS and OSPF protocols [9].

In most routing algorithms, the metric used for determining shortest paths is delay. In typical implementations, the delay estimates are based on the averaged values of either the transmission queue lengths or the delays incurred by each of the packets at this node, measured and averaged over a period of time. The problem is that these delay values are transient in nature and have inherent uncertainty. Any change in the status of the network is not immediately known to all nodes in the network. Typically, the information available at the nodes at any time is out-of-date. By the time an update is constructed, transmitted, and received, its information is already old. The update is then processed by the receiving node to compute revised Route Tables. Thus the Route Tables are based on old information and may not really reflect the best paths available at the current time. The information is even older by the time it is used to make routing decisions for individual packets. Since delays depend on queue lengths which in turn depend on the traffic, even small variations in the traffic can translate into changes in the optimal routes. Typical intervals for transmitting the periodic updates range between 1 second and 10 seconds. In terms of changes in traffic patterns, these are large intervals.

Moreover, a single crisp number does not carry sufficient information about the delay estimates in the network. We need to provide the routing algorithms with more information regarding the variation in the delays. In particular, we need to retain the confidence in the information that is being propagated away from the source. What is needed is an interval of confidence in which the delays along the particular path vary. Since a look-ahead is not possible, we base the estimate of the interval of confidence on the past history and associate levels of presumption for the information in the interval of confidence. Fuzzy numbers provide us with an easy representation of this interval of confidence with the associated level of presumption.

In the next section, we present a representation for delay using fuzzy numbers. As a test for this representation, we take one distance-vector algorithm, the distributed Bellman-Ford algorithm, and modify it to use the fuzzy delay representation. In section 4, we describe results of a simulation study that compares the original algorithm with the modified algorithm to judge the effect of using fuzzy delay estimates.

3 Fuzzy Delay Representation

The distributed Bellman-Ford algorithm is described in a graph-theoretic context.

Given a graph G = (V,E), where V denotes the set of vertices and E denotes the set of edges, and a vertex $s \in$ V, the algorithm finds the shortest paths from the given node s to every other node in the network. A non-negative length d_{ij} is associated with every edge (i, j). The algorithm first finds the shortest paths to s with at most one edge, then with at most two edges and so on. A path contains at most $(N-1)$ links where N is the number of nodes in the network. Let D_i be the shortest path length from node s to node i, at convergence, and $N(i)$ denote the set of neighbors of node i. The algorithm converges in at most $N-1$ iterations and is given by

$$\begin{aligned} D_s &= 0, \\ D_i &= min_{j \in N(i)} \, [D_j + d_{ij}], \; \forall \; i \neq s. \end{aligned} \qquad (1)$$

These equations state that the shortest path length from node s to node i is the sum of the shortest path length from node s to some node j adjacent to node i and the weight of the edge (i, j). Thus, the information needed by node s is the distance $d_{ij}, j \in N(i)$, i.e., the link distances to every neighbor, and D_j, the estimated shortest paths from the neighbor j. The node s calculates the shortest path to every other node in the network using the distance from itself to all its neighbors and the estimated shortest paths D_j obtained from its neighbors through periodic updates. The updates from a particular node are sent out asynchronously with respect to the updates generated by the other nodes. The major advantage of this algorithm is that the initial conditions can be arbitrary nonnegative numbers. This facilitates the execution of the algorithm at each of the nodes repeatedly with changes in the network status without re-initiation of the algorithm.

We use the Bellman-Ford algorithm described above as the basis for our implementation of the fuzzy set routing algorithms. In the Fuzzy Set algorithms, every node maintains the following set of data structures for routing :

- An *adjacent node table* containing the identities of the adjacent nodes and the links connecting to each of them.
- A *route table* which contains the next node along the shortest delay path to every other node in the network.

- A *delay table* which contains the delay estimates of the shortest routing path from the adjacent node which is the next node along the shortest path to a particular destination.

At every node, we monitor the queue lengths along every out-going link for a period of time which ends with the transmission of an update. During this period, we maintain the information regarding the minimum and the maximum lengths of the queues; this is the interval of confidence. At the end of each measurement period, these values are reset to the current queue length. The current queue length gets the highest presumption of 1 because it is the most up-to-date information regarding the delays along this link at this node. The tuple (minimum queue length, current queue length, maximum queue length) forms a TFN which is used as the basis of delay representation and estimation. The entries in the delay table are TFN's which represent the tuple (minimum delay, most possible delay, maximum delay) for every routing path.

During an update, every node sends its route table to all its neighbor nodes which use Equation 1 to estimate the shortest path lengths. Since at every stage, the shortest path lengths D_j are chosen, the algorithm guarantees that when it converges, the path length from s to i is the shortest [1].

Using the above mentioned algorithm and the data structures, the routing procedures for delay measurement, updating policy and the routing path calculation that have been developed using the concepts of Fuzzy Sets are presented below.

Delay measurement : The delay measurement is straightforward. The transmission queues are continuously monitored and the minimum and the maximum lengths of the transmission queues of each of the outgoing links within an update period are always known. At every packet addition or removal from a queue, the current queue length is weight-averaged with the preceding queue length so that some form of past history is incorporated into the queue lengths which are used for delay estimates. Further, the minimum and the maximum queue lengths of an update period are weight-averaged with those of the preceding update period. If the minimum (maximum) queue length is greater (lesser) than the current averaged queue length, it gets the value of the current queue length. Each of these aggregated values is multiplied with the mean packet transmission delay, the result being the queuing delay along a particular link. It should be noted here that the measurement periods are randomly phased because synchronized measurements could theoretically lead to instabilities.

Updating policy : A node generates an update periodically. When sending an update, the delay table that is sent to the adjacent nodes, reflecting the path delays to every destination in the network from that node, is built by adding the current delay table entry for every destination to the estimates of the current queuing delays along the link to the next node on the shortest path to that destination. Thus, an update carries the delay table of the node which contains the estimates of the shortest path delays from that node to every other node in the network. This update is transmitted to every adjacent node. Since the emphasis here was on the delay representation and measurement and routing path calculation, we do not elaborate on any update propagation or path-failure recovery schemes. A simple, reliable update propagation scheme was assumed.

Routing path calculation : The enhancements to the asynchronous distributed Bellman-Ford algorithm for the routing path calculation make use of the comparison criteria for delays represented as 3-tuples, corresponding to the TFN's, illustrated in Figure 1.

When a node receives an update from an adjacent node, it recomputes the routing paths, actually the next node along the shortest path for every destination, by comparing the existing paths with the one through the node from which the update was received. To every entry in the current delay table and the delay table received from the next node, we add the link propagation delay, the mean packet transmission delay and the estimate of the queuing delay along the link connecting the next node in the respective paths. The estimates of the queuing delays are obtained by multiplying the mean packet transmission time with the averaged values of the queue lengths. The fuzzy weighted means of both the delays (TFNs) are computed using an ordering method proposed by Ronald Yager [12] for the fuzzy subsets of the unit interval I which is easily extended to TFNs by reducing the TFN to unit interval through normalization. The two delays, current delay CD = (min delay, weighted mean, max delay) and new delay ND = (min delay, weighted mean, max delay), corresponding to the next node on the current shortest path and the node from which the update was received, are doubled so that there is a bias towards the shortest path, to avoid oscillation of the routing paths. The delay values thus computed are used for comparisons shown in Figure 1.

The comparison algorithm represented graphically by Figure 1 enforces subjective judgements about the delay paths on the assumption that just one comparison is not good enough to select the optimal delay

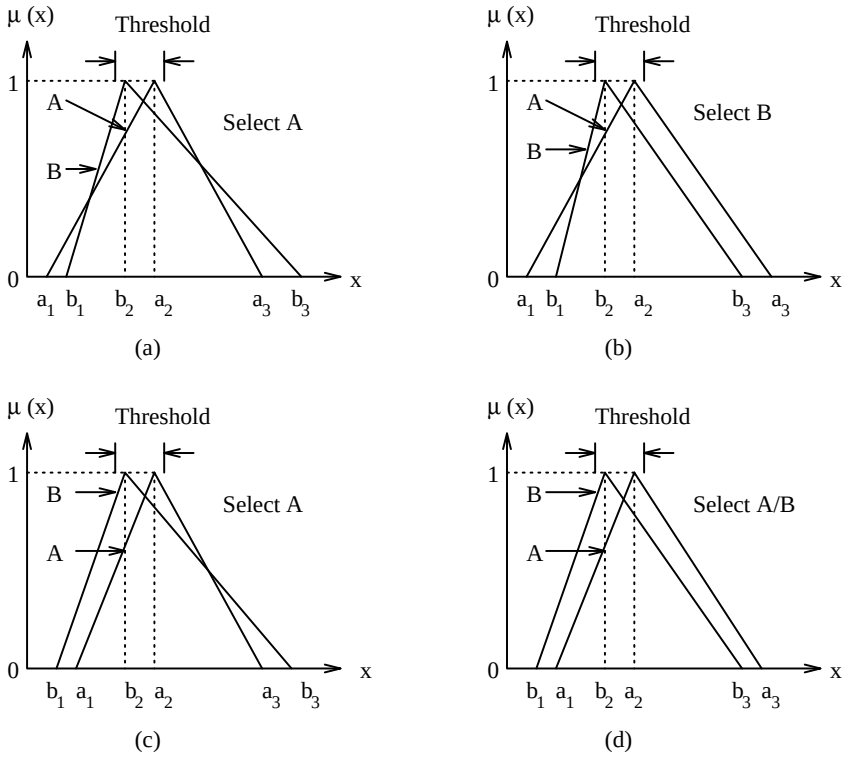

Figure 1: Comparison of Fuzzy Delays

path. The shortest delay path may not be the most optimal path in the immediate future as there could have been significant changes in the traffic since the information started propagating to the other nodes from the source. So, if there is a promising alternate path, not only in terms of delay but also in terms of congestion at the nodes, that node is considered for selecting the quasi-optimal delay path. The width of the interval spanning the fuzzy number gives us an estimate of the delay and congestion at the nodes. This information is used in the comparison criteria.

If $A = (a_1, a_2, a_3)$ and $B = (b_1, b_2, b_3)$ are estimates of the delays along two routing paths, a_2 and b_2 are compared first. If their difference is beyond a threshold value, the minimum of the two is chosen and the corresponding node is entered into the route table. The threshold is the amount of extra delay we would tolerate along the alternate path so that the delay along the alternate path would result in better overall delays. If the difference is within the threshold, there are four cases as shown in Figure 1, where there is a possibility of choosing a quasi-optimal routing path for a better overall delay performance.

- If $a_1 \leq b_1$, there are two subcases. In the first, when $a_3 \leq b_3$, A is chosen as the optimal routing path, as shown in Figure 1(a), because there is a higher possibility that the path with delay A has a smaller delay than the path with delay B. In a regular comparison method, B would be chosen to be the shorter delay. When $a_3 > b_3$ as in Figure 1(b), path with delay B is chosen, because of higher uncertainty in the interval of confidence associated with A.

- If $a_1 - b_1 \leq threshold$, there are two possible subcases again. First, if $a_3 \leq b_3$, A is chosen to be the shorter delay as shown in Figure 1(c), contrary to choosing B in the regular method. In the second case when $a_3 - b_3 \leq threshold$, there is an uncertainty about the most possible quasi-optimal routing path. Here, the decision is to retain the current delay path corresponding to A or B as in Figure 1(d).

In the rest of the cases, the comparisons are handled as in the conventional methods.

These subjective criteria help in handling the uncertainty arising from incomplete information, when the information is not up-to-date and the confidence in it is not high.

4 Simulation Results

The routing procedures presented in the previous section have been implemented in context of the Asynchronous Distributed Bellman-Ford algorithm and simulated using OPNET, a graphical network simulation tool running under Sunview on Sun workstations.

A variety of simulations were run with varying loads and different thresholds in the cases where the fuzzy comparisons developed in the previous section were used. The simulations were run long enough to give 95% confidence in the average network-wide delays with a tolerance limit of $\pm$ 5% in most cases.

The different types of simulations that were run use none, one or more of the queue length averages, the weighted mean and the fuzzy comparison procedures developed in section 3. The variations of the routing algorithm that have been simulated are as follows:

Algorithm A. Fuzzy Comparisons with Weighted Mean:

In this algorithm, all the procedures described in section 3 are implemented. The queue lengths are averaged with their corresponding previous values, the weighted mean is calculated using Yager's ordering method and the comparison criteria in Figure 1 are used in routing.

Algorithm B. Fuzzy Comparisons without Weighted Mean:

This is same as algorithm A except that the weighted mean is not computed. The comparisons use the delay values that are obtained from the averaged queue values without computing the weighted mean.

Algorithm C. Crisp Method with averaged queue values:

This is the traditional non-fuzzy algorithm and same as Algorithm B without the use of fuzzy comparison criteria.

Algorithm D. Crisp Method:

This is a simple algorithm which just takes the current queue lengths without averaging them with the preceding value and makes a simple comparison. The weighted mean is not computed and no comparison criteria are used. This method was chosen to allow a more meaningful comparison between the fuzzy and the crisp methods.

Algorithm E. Fuzzy Weighted Mean with averaged queue values:

In this algorithm, no comparison criteria are used, but it is otherwise identical to Algorithm A. This method is used for studying the effect of the comparison criteria on the delay performance.

Algorithm F. Fuzzy Weighted Mean without averaged queue values:

In this algorithm, only the weighted means are used without any fuzzy comparisons or averaging of the queue values. This method was chosen to study the effect of the fuzzy weighted mean on the delay performance.

The simulation experiments were run on a number of sample network configurations. These networks are characterized by a minimum degree of 2 and the existence of alternate paths for every source-destination pair. The simulations were run for a variety of loads ranging from 50 arrivals/sec per node to 300 arrivals/sec per node. The performance of routing algorithms A to F was tested for these loads and by varying fuzzy comparison threshold in the appropriate cases. In each of the runs, statistics regarding the delays for every source destination pair were collected. The statistics regarding the packets generated for every destination from every source in the network and the packets delivered to every destination from every source were also obtained. Further, the average network-wide delay for all the packets was collected by taking the averaged delay samples of every 800 packets.

Figure 2 shows the performance of the algorithms A to F when simulated on one of the sample networks. The delays for the lower loads for various algorithms is comparable in this network but as the offered load increases, we see that the algorithms C, D and F reach the unstable region very fast. Algorithm B has a higher operating point but decays very quickly beyond it. Algorithms A and E are the best and comparable in performance. The threshold of 3 msecs is about 25% of the longest shortest path in this network.

Our results were similar on all the sample networks on which we ran the simulations. In all cases, the algorithms using the fuzzy set delay representation performed better than the conventional algorithms using

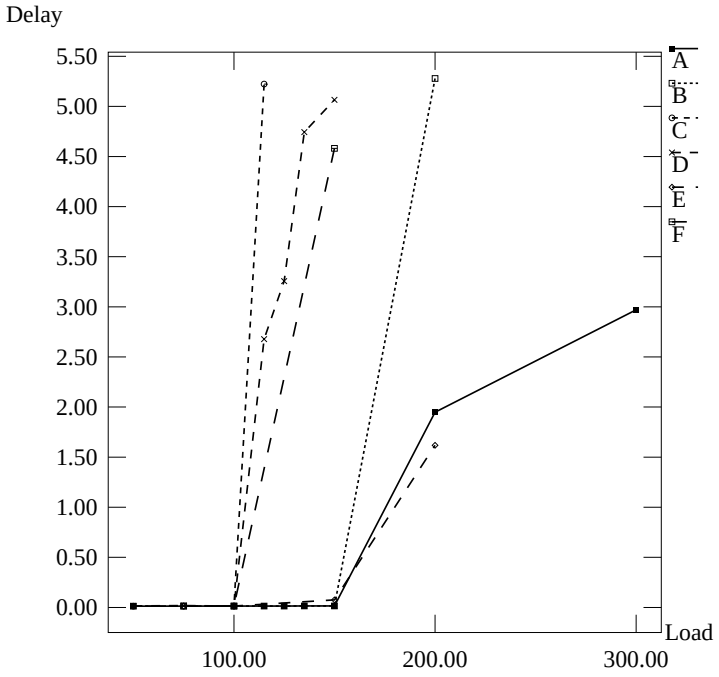

Figure 2: Delay (secs) vs. Offered Load (in packets/sec per node)

crisp delay representation. We also found that the algorithms using fuzzy comparisons and fuzzy weighted means all performed equally well. In some networks, the fuzzy weighted means seemed to have more effect on the performance than the fuzzy comparisons which seemed to have negligible effect on the delay performance. But in dense networks with many alternate paths for the various source-destination pairs, the opposite was true.

From the simulation results, it can be inferred that the routing procedures presented in section 3 have a significant impact on the delay performance of a routing algorithm. When the comparison criteria are used with the weighted means, the network-wide average delay is better or as good as the other cases at low loads and is significantly better than that of the other cases at higher loads.

5 Conclusions

In this paper, we have presented a new approach for the routing path delay estimation and manipulation for routing in wide area networks. We have used fuzzy numbers for representing the uncertainty in the delay values and have developed fuzzy comparison criteria to use this uncertainty in making routing path decisions. The routing procedures that have been developed have been simulated in the context of the distributed asynchronous Bellman-Ford algorithm. From the simulation experiments, we have seen that the fuzzy weighted mean method and the fuzzy comparison criteria result in good delay performance of the routing algorithms. The new method resulted in better or comparable delays to the traditional (crisp) methods at lower loads and in superior delay performance at higher loads. Furthermore, the use of fuzzy comparison criteria improved the delays more in the dense networks than in the lightly connected networks. In this work, the looping aspects of the routing algorithms have not been dealt with because the emphasis has been on the effects of the fuzzy decision methods rather than on developing a new routing algorithm. A possible future extension to this work would be to develop some criteria using the fuzzy set approach to handle the looping problem in routing. Overall, the work presented here has given us insight on how to handle the uncertainty regarding delays in network routing. We think this approach will be conducive to productive exploration of this aspect of network routing.

References

[1] D. P. Bertsekas and R. G. Gallager. *Data Networks*. Prentice-Hall, Englewood Cliffs, NJ, 1987.

[2] D. Dubois and H. Prade. *Fuzzy Sets and Systems: Theory and Applications*. Academic Press, NY, 1980.

[3] J. J. Garcia-Luna-Aceves. A minimum-hop routing algorithm based on distributed information. *Computer Networks and ISDN Systems*, 16:367–382, 1988-89.

[4] D. W. Glazier and C. Tropper. A new metric for dynamic routing algorithm. *IEEE Transactions on Communications*, 38(3):360–367, March 1990.

[5] J. M. Jaffe and F. H. Moss. A responsive distributed algorithm for computer networks. *IEEE Transactions on Communications*, COM-30(7):1758–1762, July 1982.

[6] A. Kaufmann and M. M. Gupta. *Introduction to Fuzzy Arithmetic : Theory and Applications*. Van Nostrand Reinhold Company, NY, 1985.

[7] G. J. Klir and T. A. Folgers. *Fuzzy Sets, Uncertainty, and Information*. Prentice-Hall, Englewood Cliffs, NJ, 1988.

[8] P. M. Merlin and A. Segall. A failsafe distributed routing protocol. *IEEE Transactions on Communications*, COM-27:1280–1287, 1979.

[9] R. Perlman. A comparison between two routing protocols: OSPF and IS-IS. *IEEE Network*, 5(5):18–24, September 1991.

[10] K. G. Shin and M. Chen. Performance analysis of distributed routing strategies free of ping-pong-type looping. *IEEE Transactions on Communications*, C-36(2):129–137, February 1987.

[11] A. S. Tanenbaum. *Computer Networks*. Prentice-Hall, Englewood Cliffs, NJ, 1988.

[12] R. R. Yager. A procedure for ordering fuzzy subsets of unit interval. *Information Sciences*, 24:143–161, 1981.

[13] L. A. Zadeh. Fuzzy Sets. *Information and Control*, 8:338–353, 1965.

[14] W.T. Zaumen and J. J. Garcia-Luna-Aceves. Dynamics of distributed shortest-path routing algorithms. In *Proc. ACM SIGCOMM-91 Conference on Communications Architectures and Protocols*, pages 31–42, Zurich, Switzerland, September 1991.