A DISTRIBUTED ALGORITHM FOR MULTIPATH COMPUTATION

SRINIVAS VUTUKURY
vutukury@cse.ucsc.edu
Computer Sciences Department
University of California
Santa Cruz, CA 95064

J.J. GARCIA-LUNA-ACEVES
jj@cse.ucsc.edu
Computer Engineering Department
University of California
Santa Cruz, California 95064
Networking and Security Center
Sun Microsystems Laboratories
Palo Alto, California 94303

Abstract—**Today's Internet routing protocols either provide a single path between each source-destination pair, or multiple paths of equal length. Furthermore, the paths provided by RIP and OSPF are not free of loops during times of network transition. Single-path routing algorithms are inherently slow in responding to congestion and temporary traffic bursts; consequently, the delays experienced by packets in these networks are far from optimal. Recently, we developed a framework for designing routing algorithms that offer "near-optimal" delays; a key component in this framework consists of using a fast responsive routing protocol that builds multipaths for each destination in the computer network, such that they are loop-free at all times. This paper studies the performance of MPATH (multipath routing algorithm) by simulation and compares it against the performance of other state-of-the-art routing algorithms.**

I. INTRODUCTION

The delays experienced in the networks that use routing algorithms such as RIP[10] and EIGRP[1] are far from optimal, because these algorithms provide only single path between each source-destination pair for packet forwarding. OSPF[14] allows a router to choose from more than one path to the same destination only when multiple paths of minimum cost exit, which means the full connectivity of the network is still not used for packet forwarding. To realize minimum-delay routing[4], the packet forwarding tables must represent a directed-acyclic graph instead of a tree, with the destination as the sink node.

Recently, we described a practical framework to obtain "near-optimal" delays [21] in dynamic networks. A key component of this framework is a fast responsive routing algorithm that computes multiple path between each source-destination pair, such that they need not all have the same length and are loop-free at every instant — in steady state as well as during network transitions. By load-balancing traffic over these multiple paths, congestion can be reduced and delays can be significantly decreased. Unfortunately, most routing algorithms do not build loop-free multiple paths. In [8], we performed load-balancing over loop-free multipath computed using the distance-vector routing algorithm DASM [22], and reduces delays dramatically. In [21], we presented the first link-state algorithm that computes loop-free multipaths and used it to implement the near-optimal framework. Recently, we introduced a new loop-free multipath routing algorithm called MPATH [19] and proved its correctness and analyzed its complexity. In this paper, we continue the study of MPATH by comparing its control message overhead and convergence time through simulations with two other routing algorithms.

This work was supported in part by the Defense Advanced Research Projects Agency (DARPA) under grants F30602-97-1-0291 and F19628-96-C-0038.

The paper is organized as follows. Section II describes the MPATH routing algorithm. Section III gives a survey of various algorithms and provides a comparative study with respect to MPATH. The simulation results and comparisons are presented in Section IV. Section V provides concluding remarks.

II. MPATH ROUTING ALGORITHM

This section illustrates the main concepts in the routing algorithm MPATH. For detailed formal description, correctness proofs and analysis, the reader is referred to [19] and [21]. The basic approach consists of nodes first computing shortest distances to destinations and then using the distances along with certain constraints (which we call loop-free invariants) to obtain a loop-free routing graph for each destination. Let N represent the nodes in the network and N^i be the set of neighbors of an arbitrary node i, and let S_j^i denote the set of next-hop choices (or successors) at node i for forwarding packets destined to node j. Then the goal of MPATH is to maintain the routing graph denoted by the link set $SG_j = \{(m,n) | n \in S_j^m,\ m \in N\}$ in presence of changing link costs, such that it is a directed acyclic graph at every instant. In contrast, the successor sets in OSPF is defined as $S_j^i = \{k \in N^i \mid D_j^k + l_k^i = D_j^i\}$, where D_j^i and D_j^k are the shortest distances of nodes i and k to j, and l_k^i is cost of link (i,k). In single-path routing protocols (e.g., RIP and EIGRP), the set S_j^i is restricted to at most one member, indicating that the routing graph SG_j is a sink-tree rooted at j. The key idea is to generalize the shortest-path trees to *shortest-multipaths*; that is, $S_j^i = \{k \in N^i | D_j^k + l_k^i = D_j^i\}$

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
1999		**00-00-1999 to 00-00-1999**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
A Distributed Algorithm for Multipath Computation	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of California at Santa Cruz,Department of Computer Engineering,Santa Cruz,CA,95064**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**		**5**	

is extended to $S_j^i = \{k \in N^i | D_j^k < D_j^i\}$. Fig. 1(A) shows an example of a network with shortest-path tree for c in Fig. 1(B) and the shortest-multipath in Fig. 1(C).

For building shortest multipaths, we require a distributed algorithm to compute shortest distances for each source-destination pairs. Though there are many algorithms for shortest-path computation, we present a new shortest-path algorithm. In link-state algorithms such as OSPF, each router stores and forwards costs of all links in the network. It was shown in LVA [6] and MPDA [21] that it is sufficient for each node to store the costs of only those links that are on the shortest path tree. Similarly in MPATH, a node stores costs of links on its shortest path tree and a copy of each neighbor's shortest-path tree. However, instead of communicating link costs directly as in MPDA, nodes in MPATH exchange distances to destinations along with the address of the second-to-last, or predecessor, node on the shortest path to the destination like in LPA [7]. MPATH translates this information to link costs and internally works with links rather than distances, and when changes to topologies have to be reported, the internally represented topology is translated back to distances and predecessors. In Fig.1(B), for example, b is the predecessor node on the shortest path from a to c. Using distances and predecessors for all nodes a, b, c and d, the complete tree can be easily constructed at the receiving neighbor. Fig. 2 shows information maintained at two arbitrary adjacent nodes i and k. T^i is the shortest path tree of node i and T_k^i is the copy of the shortest path tree T^k of neighbor k. Similarly at k, T_i^k is a copy of the tree T^i. The basic distributed shortest-path algorithm is simple and is as follows. Each node i repeatedly executes the following two steps until there are no more changes to its shortest-path tree T^i.

1. Construct the node's shortest path tree T^i from the costs of the adjacent links and the shortest-path trees T_k^i reported by the neighbors $k \in N^i$.
2. Report the new shortest path tree T^i to all its neighbors using distances and predecessor information.

Note that a node always has the current cost of the adjacent link costs in step (1). However, the node has no way to validate the costs of non-adjacent links reported by the neighbors. Therefore, the node trusts the link-cost reported by a neighbor as long as it is not an adjacent link and does not conflict with the cost reported by another neighbor for the same link. If two neighbors report the cost of the same link which are different, the node must resolve the conflict and choose the cost it considers most accurate. The following conflict resolution rule is used.

> *"If two or more neighbors report conflicting information regarding the same link, then the node must believe the neighbor that offers the shortest distance to the head of the node."*

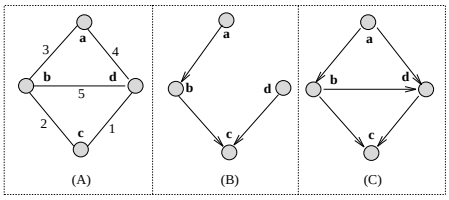

Fig. 1. Shortest-path tree and Shortest multipath

Fig. (2) visually illustrates the resolution rule. Assume the link (m, n) is reported by two neighbors p and q with different link costs. Then the cost of the link (m, n) reported by p is used to update the main table if the distance $D_m^p + l_p^i$ is less than $D_m^q + l_q^i$. The remarkable fact is that this greedy approach to link-cost validation is sufficient for purging outdated link information from the network. We generalize that rule to links of any costs and apply it in the context of propagating partial topologies. After merging all the neighbor trees into a composite graph, the shortest-path algorithm is run on the graph to obtain the tree T^i. The proof that the topology T^i at each node i converges to the correct shortest-path tree is presented in [21]. Similar resolution rule is used in SPTA[2] in the context of unit link costs and broadcasting complete topology. It is not necessary to report the complete tree as step (2) indicates; only changes to the shortest-path tree need to be reported as described in MPDA [21].

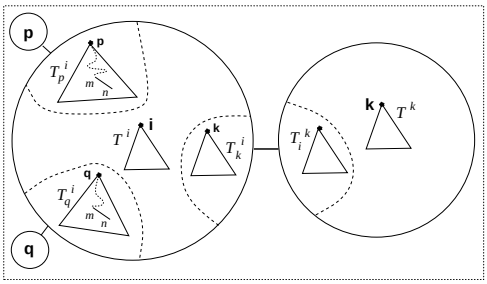

Fig. 2. Node information

Loop-free Invariants

Constructing the set $S_j^i = \{k | D_j^k < D_j^i, k \in N^i\}$ is not straight forward. The value D_j^k is not in node i's local memory. It is a value communicated (directly or indirectly) by the neighbor k and stored in the local variable D_{jk}^i in the tree T_k^i. Accordingly, node i can only compute $S_j^i = \{k | D_{jk}^i < D_j^i, k \in N^i\}$. However, because of non-zero propagation delays, the values of D_j^k and D_{jk}^i can be inconsistent during network transitions. Similarly, the copy of distance D_j^i at neighbor k, D_{ji}^k, can be inconsistent with D_j^i except in steady-state. In steady-state, assume that neighbor k is using i as a successor for destination j. Then $D_j^k = D_{jk}^i > D_j^i = D_{ji}^k$. Let D_j^i increase to a value greater

than D_j^k due to a network event. Node i should now use k as a successor, but adding k in the successor set S_j^i without restraint would result in a loop. Therefore, the neighbor k must first remove i from its successor set before i can put k in its successor set. This requires neighbor-to-neighbor synchronization. Node i first reports its new distance to the neighbor and can add k to its successor set only after k acknowledges the new distance and removes i from its successor set. To accomplish this, we use a new variable FD_j^i, called the feasible distance, to compute the successor sets. FD_j^i closely follows the value of D_j^i while always satisfying the following *loop-free invariants*.

$$\begin{aligned} FD_j^i(t) &\leq D_{ji}^k(t) \qquad k \in N^i \\ S_j^i(t) &= \{\, k \mid D_{jk}^i(t) < FD_j^i(t) \} \end{aligned}$$

Whenever D_j^i increases a neighbor-to-neighbor synchronization is used so that FD_j^i is increase to D_j^i only after the copy of D_j^i is properly updated at the neighbors. It is proved in [19] that the LFI conditions ensure loop-freedom of the successor graphs SG_j at all times. The algorithm MPATH ensures that at convergence $FD_j^i = D_j^i = D_{ji}^k$ for all i and j and $k \in N^i$, which implies that the routing graph at convergence is the required shortest-multipath.

III. Comparison with other algorithms

A. Distance-vector algorithms

RIP[10] is based on the Distributed Bellman-Ford (DBF) algorithm for computing the shortest-paths to destinations. In networks that use RIP, nodes exchange only distances to destinations and have no knowledge of the network topology, and due to lack of this information they suffer from the infamous *counting-to-infinity* problem[2]. Several techniques have been proposed to tackle this problem. DUAL[5], which is the algorithm used in EIGRP [1], uses diffusing computations [3]. In addition to DUAL, several algorithms based on distance vectors have been proposed that use diffusing computation to overcome the counting-to-infinity problem of DBF [17], [13], [12], [22]. Jaffe and Moss[12] allow nodes to participate in multiple diffusing computation of the same destination, which requires use of unbounded counters. In contrast, DUAL restricts a node to participate in only one diffusing computation at any one time for any destination. All these routing algorithms provide only one loop-free path to the destination. Zaumen and Garcia-Luna-Aceves [22] presented DASM, which is the first distance-vector algorithm that provides loop-free multipaths. DASM is shown to perform better than DUAL, which was previously the best distance-vector algorithm. All the algorithms mentioned above use diffusing computations that potentially span the whole network. In contrast, MPATH uses only single-hop synchronization, i.e., a node needs xto synchronize only with its neighbors. It is interesting to see how these synchronization mechanisms influence the convergence times. For this reason, we chose DASM as the candidate routing algorithm based on distance vectors with whih to compare MPATH.

B. Link-state algorithms

In link-state algorithms, full topology information is flooded through the network. When periodic updates are made as in the case of near-optimal routing, the overhead is very high. Routing protocols based on topology-broadcast (e.g., [18], [15]) incur too much communication overhead, which forces the network administrators to partition the network into areas connected by a backbone. This makes OSPF complex in terms of router configuration required. A couple of routing algorithms have been proposed that operate using partial topology information (LVA [6],ALP [9]) to eliminate the main limitation of topology-broadcast algorithms.

In the above link-state algorithms nodes distinguish new and old link information using sequence numbers, which are not only an added overhead but also require to be reset o occasions. Instead of sequence numbers, MPATH uses a novel update rule to distinguish old and new information. Like the link-state algorithms MPATH is free from count-to-infinity problem. In OSPF, multiple equal-cost paths are computed to each destination if they exist. However, these paths are not loop-free during network transitions, and even if short-lived, these loops may cause incorrect link-cost measurements. In contrast, MPATH maintains multiple paths that need not be of equal cost and which are loop-free at every instant.

Several distributed shortest-path algorithms [11], [16], [7] have been proposed that use the distance and second-to-last hop to destinations as the routing information exchanged among nodes. These algorithms are often called path-finding algorithms or source-tracing algorithms. Though they exchange distances like the distance-vector algorithms, they are akin to link-state algorithms because they internally maintain path information obtained using the predecessor information; distance-vector algorithm have no knowledge of network topology. These algorithms eliminate DBF's counting to infinity problem using the path information. Some of them [7] are more efficient that any of the routing algorithms based on link-state information proposed to date. Furthermore, LPA [7] is loop-free at every instant, but provides only one path. MPATH is the first path-finding algorithm that builds multiple loop-free paths. As in LPA, the synchronization in MPATH is geared towards providing loop-free paths.

IV. Simulation results

The simulations compare the control overhead and convergence times of MPATH, topology broadcast and DASM. The reason for choosing topology broadcast is that it is the

approach used in OSPF, for which commercial implementations exist and it provides multiple paths of equal length. Its convergence time is fairly constant and depends on the diameter of the network. Ideally, MPATH should approach the convergence times of topology broadcast, that is, the extra time needed to enforce loop-freedom should be negligible. We expect MPATH to have far less message overhead, because of its reliance on only partial topology information. On the other hand, DASM is the only prior distance-vector routing algorithm that provides loop-free multipaths to each destination, and has been shown to be more efficient than DUAL, which is used in EIGRP. DASM achieves loop-freedom through diffusing computations that span the whole network. In contrast, MPATH uses only neighbor-to-neighbor synchronization. It is interesting to see how convergence times are effected by the synchronization mechanisms. Also, it is not obvious how the control message overheads of DASM and MPATH compare.

The performance metrics used for comparison are the control message overhead and the convergence times. The simulator is an event-driven real-time simulator called CPT. Simulations are performed on the CAIRN topology, which was also used in [20]. The topology is flat and we do not use area aggregations in the simulations. There is no reason to believe that the presence of areas would favor one routing algorithm over others.

Two types of events are triggered in the network: link-status changes and link-cost changes. Link failures and link recovery events are classified as link-status changes. In practice, links and nodes are highly reliable and change status much less frequently than link costs which are a function of the traffic on the link. We do not simulate node failures because of the problems resulting due to loss of sequence numbers by the nodes, which only effect the functioning of topology broadcast here. Special reset protocols that discover sequence numbers should be implemented for topology broadcasting based on sequence numbers.

We also restrict link-status changes to a single change; that is, only one link failure or link recovery can occur at any time during the measurement interval. Because the links and nodes in the network are highly reliable, simultaneous multiple topological changes are much less likely to occur and it is reasonable to assume that tables converge between topological changes. However, link costs of multiple links can change simultaneously and repeatedly before the tables converge to the latest costs. This is the case when near-optimal delay routing of [21] is used, in which the link costs are periodically measured and reported. For these reasons, we simulate only single link-status changes and multiple link-cost changes.

Link-status changes: Each link in turn is made to fail and then recover, and the control message overhead and convergence times are measured in each case. The worst-case and the averages of control message overhead and convergence times for link failures and link recoveries are given in Table 1. Figs. (3)-(6) give the performance figures for each event. For brevity, the performance curves for topology broadcast are labeled "TOPB". For link failures and recoveries MPATH has lower average message overhead than TOPB, which is due to the use of partial topologies in MPATH, compared to full topologies in TOPB. However, MPATH incurs a higher worst-case message overhead than TOPB, because of the synchronization used in MPATH to provide loop-freedom. MPATH has larger overhead (in bytes) than DASM under link recoveries, because neither invokes synchronization, but MPATH exchanges predecessor information in addition to distances. However, DASM requires more messages under link-failures because of the multihop synchronization that DASM uses. The same argument can be applied for the convergence times.

TABLE 1

Control messages (bytes)			
	Worst-case	Avg	Std-dev
Link failures			
TOPB	555.00	555.00	0.00
DASM	3312.00	1052.70	792.19
MPATH	1160.00	443.29	266.06
Link recoveries			
TOPB	552	552	552
DASM	1120.0	353.41	266.43
MPATH	944	423.52	230.95
Link-cost changes			
TOPB	9384.00	9384.00	0.00
DASM	11520.00	10050.93	742.10
MPATH	6856.00	5272.53	702.51
Convergence times (ms)			
	Worst-case	Avg	Std-dev
Link failures			
TOPB	1.46	1.20	0.14
DASM	3.30	2.16	0.78
MPATH	2.02	1.11	0.42
Link recoveries			
TOPB	1.46	1.20	0.14
DASM	1.48	0.97	0.39
MPATH	1.52	1.08	0.37
Link-cost changes			
TOPB	5.48	5.48	0.00
DASM	9.82	7.75	0.71
MPATH	6.46	4.87	0.77

Multiple link-cost changes: When near-optimal routing framework is implemented, multiple links change costs. To study the protocol overhead under such scenarios, multiple simultaneous link costs are changed and the performance is measured. Link costs are chosen randomly within a range. The average message overhead and convergence times are shown in the Table 1. MPATH has lower worst-case and average message overhead than TOPB and DASM. MPATH has lower worst-case and average convergence time than DASM. The average convergence time for MPATH is also

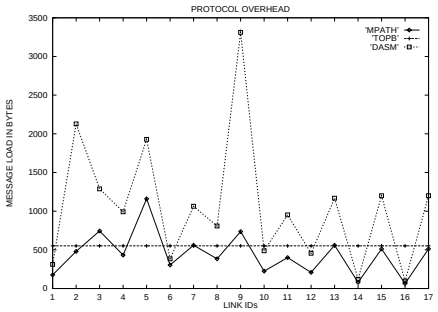

Fig. 3. Link failures. Message overhead

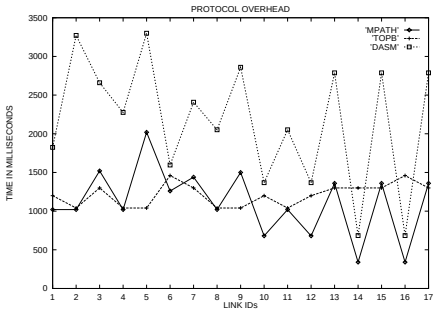

Fig. 4. Link failures. Convergence times

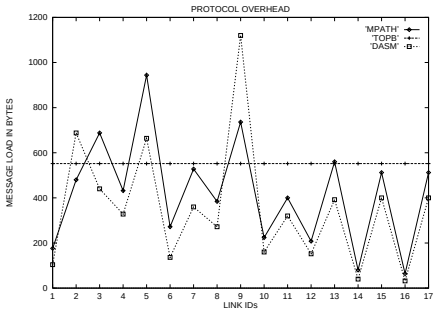

Fig. 5. Link recoveries. Message overhead

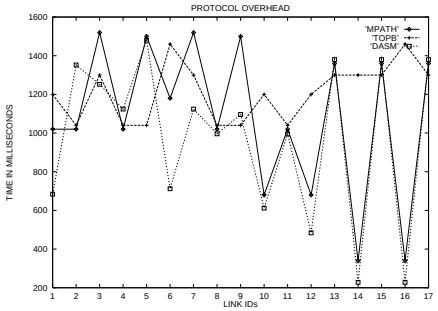

Fig. 6. Link recoveries. Convergence times

lower than TOPB. Only in the case of worst-case convergence time, MPATH showed a higher value than in TOPB, which is again due to synchronization used in MPATH.

V. Concluding Remarks

The performance of a new loop-free multipath routing algorithm called MPATH [19] is compared with two state-of-the-art routing algorithms. The algorithm has low control message overhead because of its reliance on reporting only partial topology information. By virtue of its one-hop synchronization, the convergence times of the algorithm are better than those of distance-vector routing algorithms that use diffusing computation that potentially span the entire network. The multiple successors at a node made available by the algorithm can be used for traffic load-balancing and can prove valuable in the internet for handling congestion and minimizing delays.

References

[1] R. Albrightson, J.J. Garcia-Luna-Aceves, and J. Boyle. EIGRP-A Fast Routing Protocol Based on Distance Vectors. *Proc. Networld/Interop 94*, May 1994.

[2] D. Bersekas and R. Gallager. *Data Networks*. Prentice-Hall, 1992.

[3] E.W.Dijkstra and C.S.Scholten. Termination Detection for Diffusing Computations. *Information Processing Letters*, 11:1–4, August 1980.

[4] R. G. Gallager. A Minimum Delay Routing Algorithm Using Distributed Computation. *IEEE Trans. Commun.*, 25:73–84, January 1977.

[5] J.J. Garcia-Luna-Aceves. Loop-Free Routing Using Diffusing Computations. *IEEE/ACM Trans. Networking*, 1:130–141, February 1993.

[6] J.J. Garcia-Luna-Aceves and J. Behrens. Distributed, scalable routing based on vectors of link states. *IEEE Journal on Selected Areas in Communications*, October 1995.

[7] J.J. Garcia-Luna-Aceves and S. Murthy. A path-finding algorithm for loop-free routing. *IEEE/ACM Trans. Networking*, February 1997.

[8] J.J. Garcia-Luna-Aceves, S. Vutukury, and W. Zaumen. A Practical Approach to Minimizing Delays in the Internet Routing Protocols. *ICC'99*, 1999.

[9] J.J. Garica-Luna-Aceves and M. Spohn. Scalable link-state internet routing. *Proc. International Conference on Network Protocols*, October 1998.

[10] C. Hendrick. Routing Information Protocol. *RFC*, 1058, june 1988.

[11] P. A. Humblet. Another Adaptive Distributed Shortest Path Algorithm. *IEEE Trans. Commun.*, 39:995–1003, June 91.

[12] J. M. Jaffe and F. H. Moss. A Responsive Distributed Routing Algorithm for Computer Networks. *IEEE Trans. Commun.*, 30:1758–1762, July 1982.

[13] P. M. Merlin and A. Segall. A Failsafe Distributed Routing Protocol. *IEEE Trans. Commun.*, 27:1280–1287, September 1979.

[14] J. Moy. OSPF Version 2. *RFC*, 1247, August 1991.

[15] R. Perlman. Fault-tolerant broadcast of routing information. *Computer Networks and ISDN*, 7, 1983.

[16] B. Rajagopalan and M. Faiman. A Responsive Distributed Shortest-Path Routing Algorithm with Autonomous Systems. *Internetworking: Research and Experience*, 2:51–69, March 1991.

[17] A. Segall. Optimal distributed routing for virtual line-switched data networks. *IEEE Trans. Commun.*, 27:201–209, January 1979.

[18] J. Spinelli and R. Gallager. Event Driven Topology Broadcast without Sequence Numbers. *IEEE Trans. Commun.*, 37:468–474, 1989.

[19] S. Vutukury and J.J. Garcia-Luna-Aceves. An algorithm for multipath computation using distance-vectors with predecessor information. *Proc. of ICCCN*, Oct. 1999.

[20] S. Vutukury and J.J. Garcia-Luna-Aceves. A Scalable Architecture for Providing Deterministic Guarantees. *Proc. of ICCCN*, Oct. 1999.

[21] S. Vutukury and J.J. Garcia-Luna-Aceves. A Simple Approximation to Minimum Delay Routing. *Proc. of ACM SIGCOMM*, Sept. 1999.

[22] W. T. Zaumen and J.J. Garcia-Luna-Aceves. Loop-Free Multipath Routing Using Generalized Diffusing Computations. *Proc. IEEE INFOCOM*, March 1998.