

# Loop-Free Multipath Routing Using Generalized Diffusing Computations

William T. Zaumen  
Sun Microsystems, Inc.  
901 San Antonio Avenue  
Palo Alto, CA 94043

J.J. Garcia-Luna-Aceves  
Computer Engineering Department  
School of Engineering  
University of California  
Santa Cruz, CA 95064

## Abstract

*A new distributed algorithm for the dynamic computation of multiple loop-free paths from source to destination in a computer network or internet are presented, validated, and analyzed. According to this algorithm, which is called DASM (Diffusing Algorithm for Shortest Multipath), each router maintains a set of entries for each destination in its routing table, and each such entry consists of a set of tuples specifying the next router and distance in a loop-free path to the destination. DASM guarantees instantaneous loop freedom of multipath routing tables by means of a generalization of Dijkstra and Scholten's diffusing computations. With generalized diffusing computations, a node in a directed acyclic graph (DAG) defined for a given destination has multiple next nodes in the DAG and is able to modify the DAG without creating a directed loop. DASM is shown to be loop-free at every instant, and its average performance is analyzed by simulation and compared against an ideal link-state algorithm and the Diffusing Update Algorithm (DUAL).*

## 1. Introduction

The routing protocols used in today's computer networks and internetworks are based on shortest-path algorithms that are usually classified as distance-vector algorithms and link-state or topology-broadcast algorithms. In a distance-vector algorithm, a router knows the length of the shortest path from each neighbor router to every network destination, and uses this information to compute the shortest path and next router in the path to each destination. A router sends update messages to its neighbors—and only to them; each update message contains a vector of one or more entries, each of which specifies, as a minimum, the distance to a given destination. Some distance-vector algorithms and protocols also specify the second to last hop in the shortest path [7] [9] or the entire path to a destination [12]. In a link-state algorithm, each router broadcasts messages containing the state of each of the router's adjacent links to every other router in the network; routers use this information to compute shortest paths to all network destinations. Recently, Garcia-Luna-Aceves and Behrens introduced a hybrid type of routing algorithm, called link-vector algorithms [6], in which a router communicates to its neighbors the costs of those links that belong to its preferred paths to reach destinations.

Data packets can be routed using either incremental routing or source routing. With incremental routing, also called destination-based or hop-by-hop routing, each router in the path to the destination makes a decision of where to route the packet next. In contrast, with source routing, the source router specifies the entire path to the destination in the header of each data packet, or a connection is established so that only the connection-establishment packet has to specify the entire path, while the rest of the packets need only specify the identifier of the connection. Using incremental routing to route packets along multiple loop-free paths can be more desirable than using source routing, because the routers forwarding a

packet can react more quickly to changes in local network conditions than the source router can. Furthermore, source routing limits the number of different paths that can be used to those known by the source, which means that only a small number of possible paths can be used. In contrast, if multiple loop-free paths are obtained using incremental routing, packets can flow through paths that the source need not know. It is desirable, of course, to avoid out-of-order packet delivery, and this can be achieved with incremental routing by arranging that packets associated with particular end points (both source and destination) follow the same path. Various techniques can be used to do this; the choice of techniques depends on the hardware and software capabilities of the router, but can be done in ways that do not impact the algorithm used to compute paths to destinations, and as such are outside the scope of this paper.

The provision of multiple paths in existing internet routing protocols is very limited. For example, OSPF [15] allows a router to choose more than one path to the same destination only when multiple paths of minimum cost exist. IGRP [1] allows a router to forward packets through paths whose length is less than the product of the shortest-path length times a variance factor specified by the network administrator. If the variance is too small, this approach becomes one of routing through one or more shortest paths; if the variance is too large, long-term routing loops can occur even when all routing tables are correct. The link-state algorithm reported in [2] and [3] attempts to provide multiple loop-free paths by requiring a router to forward packets to a destination through neighbors who have reported a smaller distance than the router forwarding the packet. None of these three approaches prevents routing loops when routing tables are inconsistent. To prevent routing loops due to inconsistent routing tables, the algorithm in [3] requires every router to have consistent topology information; however, requiring that the topology tables of all routers be consistent is impractical in large internets.

Because of the routing loops that can occur with incremental routing based on any shortest-path algorithm, Gardner, et al. [8] argue that source routing be used. This paper demonstrates that incremental routing over multiple loop-free paths is possible, even when routing tables are inconsistent. It extends previous results on loop-free routing along shortest paths by introducing the concept of a *shortest multipath*, which is defined as a directed acyclic graph defined by the successor entries of the routing tables of the routers in all the paths from a source to a destination that are guaranteed to be loop-free at a given instant. A shortest multipath to a destination contains the shortest path, but may contain longer paths, as shown in Figure 1, where dark arrows show the shortest path between routers (and their successors) and destination  $j$ . Gray arrows denote part of a shortest multipath that does not lie along the shortest path from that point in the shortest multipath. The shortest multipath from router  $i$  to destination  $j$ , as shown in Figure 1,

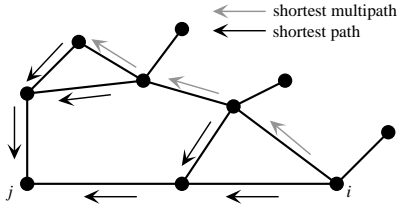


Fig. 1. Shortest Multipath Routes

contains additional paths with lengths longer than the shortest path.

In this paper, we introduce the Diffusing Algorithm for Shortest Multipath (DASM) to compute shortest multipaths distributedly. DASM is based on a generalization of the basic scheme first proposed by Dijkstra and Scholten for the dissemination of distributed operations [4], [13]. This generalization, which we call *Generalized Diffusing Computations*, permits a router to synchronize the updating of the routing-table entry for a given destination while maintaining multiple successors in loop-free paths to the destination. Although several approaches for shortest-path routing with instantaneous loop-freedom have been proposed based on Dijkstra and Scholten’s diffusing computations [10] [5] or similar approaches [14], [16], all of these algorithms guarantee instantaneous loop freedom only if the single successor chosen by each router according to the algorithm is the one used for packet forwarding. In contrast, DASM maintains instantaneous loop-freedom through multiple successors for every destination at each router; the only information exchanged among neighbor routers consist of vectors of shortest distances to destinations. Synchronizing the update activity of routers eliminates looping but can incur considerable additional overhead. To reduce this overhead, DASM uses a feasibility condition to determine when a router can update its routing table without synchronizing with other routers. This condition depends on the router’s own distance and the distances reported by its neighbors. DASM operates with arbitrary transmission or processing delays, assumes arbitrary positive link costs, and provides correct routing tables within a finite time after the occurrence of an arbitrary sequence of link-cost or topological changes.

Section 2 presents the network model and notation assumed to describe DASM. Section 3 describes the operation of DASM and shows how it supports loop-free area routing. Section 4 proves that DASM is loop-free at every instant and that it converges to correct routing tables within a finite time after a sequence of link-cost or topology changes. Sections 5 and 6 analyze DASM’s complexity and average performance; it is shown that DASM provides considerable improvements over DUAL [5] and the ideal link-state algorithm, which constitutes an upper bound on the performance of existing routing protocols based on topology broadcast (e.g., OSPF).

## 2. Network Model and Notation

An internet is modeled as an undirected connected graph in which each link has two lengths or costs associated with it—one for each direction—and in which any link of the graph exists in both directions at any one time. Each router has a unique identifier, and link costs can vary in time but are always positive. The smallest distance between two routers is defined as the sum of the link costs in any path of least cost or *shortest path* between them. In addition, DASM requires an underlying protocol, called a *neighbor-to-neighbor protocol* for passing messages to neighbors reliably and for detecting changes in link status. The requirements for this protocol are:

- Every router detects within a finite time the existence of a new neighbor or the loss of connectivity with a neighbor.

- All packets transmitted over an operational link are received correctly and in the proper sequence within a finite time.
- All messages, changes in the cost of a link, link failures, and new-neighbor notifications are processed one at a time within a finite time and in the order in which they occur.

Throughout this paper, the following notation is used:

$G(V, E)$ : a connected network of arbitrary topology in which DASM is used with  $V$  denoting the set of all routers and  $E$  denoting the set of all links.

$N^i$ : the set of routers connected through a link with router  $i$ ; a router in that set is said to be a neighbor of router  $i$ .

$k$ : a neighbor of router  $i$ .

$(i, k)$ : the link between routers  $i$  and  $k$  in  $V$ .

$l_k^i$ : the cost of the link to neighbor  $k$ ; if the link to neighbor  $k$  fails, the cost is assumed to be  $\infty$ .

$j$ : a destination router in  $V$ .

$SG_j$ : the directed graph consisting of those routers that can reach router  $j$ . Each link in  $SG_j$  consists of a directed link from a router to a link in that router’s successor set.

## 3. The Distributed Algorithm for Shortest Multipath

### 3.1 Principles of Operation

In any routing algorithm, the aggregate of each router’s routing-table entry maintained for a given destination  $j$  defines a directed graph rooted at  $j$ . When the algorithm converges to correct values, this directed graph is acyclic.

The objective of a loop-free routing algorithm is to ensure that the routing tables never contain loops, i.e., that the routing-table entries of routers define a directed acyclic graph (DAG) for each destination at every instant.

The basic objective in DASM is to permit routers to maintain at all times a DAG for each destination that has much more connectivity than the directed trees obtained with prior loop-free routing algorithms [5], [7], [10], [14]. To accomplish this, each router has a successor set for each destination, rather than a single successor.

A message sent by router  $i$  is an ordered triplet of the form [MessageType,  $j$ ,  $RD_j^i$ ], where MessageType can be “update,” “query” or “reply,” where  $j$  is a destination, and where  $RD_j^i$  is the cost reported in the message.

DASM itself does not depend on the use of sequence number or timers directly. It follows that, in DASM, a router knows only the distances to destinations reported by each neighbor, and communicates the same information to its neighbors.

Requiring routers to synchronize their update activity every time any one of them must change its routing table incurs excessive communication overhead. Accordingly, DASM uses generalized diffusing computations to ensure loop freedom only when a sufficient condition for loop-freedom is not met after an input event is received, and behaves much like the Distributed Bellman-Ford (DBF) algorithm when such a condition is met. Accordingly, each router that uses DASM can be in one of two states: *active* and *passive*.

Routers begin execution in the passive state and need to know only about their own existence; they may become active at various times by sending queries to all its neighbors. When the router receives a reply from each neighbor, it may either become passive or stay active. The time interval between sending a query and receiving the last reply to that query is called an *active phase*, and multiple active phases for a destination can occur successively, but may not overlap. A flag  $r_{jk}^i$  is maintained to ensure this behavior:  $r_{jk}^i$  is true if router  $i$  has sent a query to router  $k$  for destination  $j$  but has not yet received a reply and false otherwise.

As with DUAL [5], DASM cannot send a query during an active phase, so that once a router sends a query to its neighbors for

destination  $j$ , the same router cannot send more queries for destination  $j$ , until all replies are received. When router  $i$  sends a query for destination  $j$ , router  $i$  sets  $r_{jk}^i = \text{true}$  for all  $k \in N^i$ , and sets  $r_{jk}^i = \text{false}$  when a reply from router  $k$  is received. Router  $i$  is not allowed to send another query for destination  $j$  until  $r_{jk}^i = \text{false}$  for all  $k \in N^i$  (the time at which this first occurs after sending a query marks the end of an active phase). This prevents active phases from overlapping, which simplifies the state that each router needs to maintain for each destination.

The rest of this section describes the conditions used in DASM to determine when to invoke generalized diffusing computations, how generalized diffusing computations work in DASM in static topologies, how topology changes are handled, and why DASM supports not only multiple paths to each destination in a “flat” topology, but also loop-free multipath area routing.

### 3.2 Sufficient Conditions for Loop-Free Routing

DASM avoids routing-table loops by forcing routers to choose as their next hops to destinations only among those neighbor routers that satisfy a destination-based ordering constraint that is valid for all routers in the network. Simply put, for a given destination, a router  $a$  is given a label such that it can use a neighbor router  $b$  as a successor to the destination if and only if  $b$ 's label is strictly smaller than  $a$ 's own label. The following describes how DASM accomplishes this making use of the following variables:

$FD_j^i$ : is the feasible distance at router  $i$  for destination  $j$ .

$S_j^i$ : is the set of neighbors of router  $i$  that are used as successors of router  $i$  for destination  $j$ . This set represents the neighbors to use along all loop-free paths maintained by the routing-algorithm.

$D_{jk}^i$ : the distance reported by neighbor  $k$  in an update, query, or reply.

$D_{jk}^{i*}$ : an upper bound on neighbor  $k$ 's feasible distance.

$RD_j^i$ : the cost for destination  $j$  that will be used in messages sent to neighbors.

Let  $i \in V$  be an arbitrary router in graph  $G(V, E)$ . For each destination  $j$ , and every router  $k \in N^i$ , DASM requires that router  $i$  maintains a label  $FD_j^i$  representing a feasible distance and a label  $D_{jk}^{i*}$  that by design satisfies  $FD_j^i \leq D_{jk}^{i*}$  at all times (this is proven in Theorem 1). Because  $FD_j^i$  represents a value available at router  $k$  (which is a neighbor of router  $i$ ), and given that  $D_{jk}^{i*}$  and  $FD_j^i$  represent values available at router  $i$ , the condition  $FD_j^i \leq D_{jk}^{i*}$  implies that router  $i$  has an upper bound on each of its neighbor's feasible distances for destination  $j$ , and loop-free paths are easily obtained by requiring that the next router along a path have a feasible distance smaller than the current router's feasible distance. The following definition specifies this ordering constraint as imposed in DASM:

*Definition 1: Loop-Free Routing Condition (LRC).*

Router  $i$  can choose any neighbor router in the set  $S_j^i = \{k \in N^i \mid D_{jk}^{i*} < FD_j^i\}$ .

Clearly, for any router  $k$  in  $S_j^i$ ,  $FD_j^k < FD_j^i$ . Then, provided that all labels are properly updated, loop-free multipath routes are obtained at each instant of time by simply choosing, at each router  $i$  any router in  $S_j^i$  as the next hop. Furthermore, if  $S_j^i$  contains routers other than ones along the shortest path, there are more available paths to a destination than what is possible with shortest-path routing alone.

In order to maintain the condition  $FD_j^k < FD_j^i$  at all times, DASM behaves differently within a passive phase than within an active phase. While passive (for destination  $j$ ), a router's feasible distance must be a decreasing (as opposed to monotonically

increasing) function of time, and its feasible distance must be no larger than its actual distance to the destination. Thus, if the distance that will be reported to neighbors falls below the feasible distance, the feasible distance must be decreased. Router  $i$  can exploit this behavior as described below to maintain an upper bound on router  $k$ 's feasible distance, thus ensuring loop-free routing while routers are passive. It is also important, however, for the neighbor providing the shortest path to be in router  $i$ 's successor set, and if this is not true, router  $i$  must increase its feasible distance. This is done in two steps. First a query containing the desired distance is sent, starting an active phase. This query in effect asks permission from router  $i$ 's neighbors for router  $i$  to raise its feasible distance. When a reply has been received from each neighbor, ending the active phase (see Section 3.5 for how link failures are handled), router  $i$  can safely raise its feasible distance to the lowest value reported in any message since the query. Between receiving a query and sending a reply, an additional variable is used to track the minimum reported distance since the query was received, and this becomes the new upper bound once the reply is sent. This is described in detail below, and in subsequent sections, additional conditions on the behavior of a router during an active phases are introduced. These conditions are needed for ensuring convergence and termination.

To proceed, several additional definitions are needed:

$\tilde{D}_{jk}^{i*}$ : an upper bound on the feasible distance of neighbor  $k$  after a query from the neighbor is processed and before the next reply is sent.

$QS_j^i$ : the set of routers for which a query has been received, by router  $i$ , but for which a reply has not been sent.

The following summarizes the way in which the values of  $D_{jk}^{i*}$  and  $FD_j^i$  are updated to make LRC a valid ordering constraint.

Initially,  $S_j^i = \emptyset$ ,  $r_{jk}^i = \text{false}$ , and the values of the remaining variables are infinite, except when  $i = j$ , in which case  $FD_j^i = RD_j^i = 0$ . At router  $i \neq j$ ,  $FD_j^i \leq RD_j^i$ , and  $FD_j^i$  increases *only at the end* of an active phase, at which point  $i$  can raise  $FD_j^i$  to the minimum value of  $RD_j^i$  that occurred during the active phase.

When router  $k$  sends a message to router  $i$ , the value in the message will be the current value of  $RD_j^k$ . When router  $i$  receives this message, it sets  $D_{jk}^i$  to that value, and updates  $D_{jk}^{i*}$  so that it contains the minimum of  $D_{jk}^i$  and the previous value of  $D_{jk}^{i*}$ . If the message was a query, router  $i$  also sets  $\tilde{D}_{jk}^{i*}$  to the new value of  $D_{jk}^i$ , and subsequently sets  $\tilde{D}_{jk}^{i*}$  to the minimum of  $D_{jk}^i$  and the previous value of  $\tilde{D}_{jk}^{i*}$  upon receiving each additional message from router  $k$  until router  $i$  sends router  $k$  a reply. When the reply is sent,  $D_{jk}^{i*}$  is set to  $\tilde{D}_{jk}^{i*}$ .

Whether a reply has been sent or not is determined by using the set  $QS_j^i$ —a neighboring router  $m$  is added to this set when a query for destination  $j$  is received from router  $m$ , and router  $m$  is removed from  $QS_j^i$  when a reply for destination  $j$  is sent by router  $i$  to router  $m$ . This is illustrated in Figure 2 (although  $\tilde{D}_{jk}^{i*}$  and  $QS_j^i$  are not shown). It is worth pointing out that when router  $i$  is passive,  $D_{jk}^{i*} = \tilde{D}_{jk}^{i*}$ .

As we will show in Section 4.1, the above implies that  $FD_j^k \leq D_{jk}^{i*}$ . Therefore, because the successor set  $S_j^i$  satisfies  $S_j^i = \{k \in N^i \mid D_{jk}^{i*} < FD_j^i\}$ , then loop-free routing follows because a loop would imply that  $FD_j^i < FD_j^i$ , which is not possible.

Although LRC leads to loop-free paths at every instant, the paths obtained by simply applying LRC need not be useful—additional conditions are needed, which we discuss subsequently.

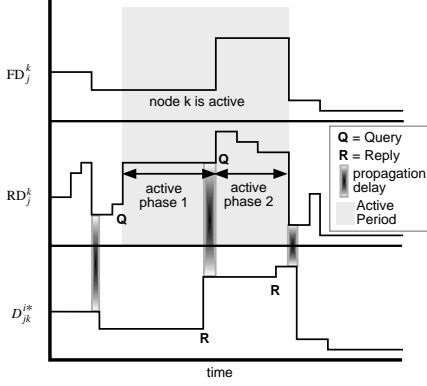


Fig. 2. Feasibility Condition and Reported Distances

### 3.3 Conditions for Local Computations

A router  $i$  is said to use a *local computation* for destination  $j$  when it updates its routing-table entry for that destination without requiring any internodal synchronization. In such a case, we say that router  $i$  is in the passive state. In passive state, a router sends only updates or replies to queries, but it does not send its own queries. Because internodal synchronization requires additional communication overhead, it is desirable for routers to execute local computations as much as possible, but without creating loops.

Because DASM is designed to provide shortest multipaths, routers should be allowed to update their successor sets (i.e., which neighbors they can use to reach destinations) without synchronizing with other routers, as long as those successor sets contain neighbors that provide shortest paths to destinations. Otherwise, the lengths of the loop-free paths obtained using LRC could increase without bound, even if they are loop-free.

Hence, a router remains passive as long as LRC yields paths that include shortest paths. The following describes how DASM accomplishes this using the following variables:

$D_{\min}^{ij}$ : the cost of a path from router  $i$  to destination  $j$  along the shortest path.  $D_{\min}^{ij} = \min\{D_{jk}^i + l_k^i \mid k \in N^i\}$ , assuming that  $D_{jk}^i$  has been updated if router  $i$  is responding to a message.

$Z_j^i$ : the set of neighbors of router  $i$  that lie along a shortest path to destination  $j$ , and is computed after  $D_{\min}^{ij}$  is updated.

$S_{jP_1}^i$ : the value the successor set of router  $i$  would have after an active-to-passive transition or if router  $i$  remains passive.

$S_{jP_1}^i = \{k \in N^i \mid \tilde{D}_{jk}^{i*} < \min(\text{FD}_j^i, D_{\min}^{ij})\}$ , and is computed after  $\tilde{D}_{jk}^{i*}$ ,  $\text{FD}_j^i$ , and  $D_{\min}^{ij}$  are updated.

In terms of the above variables, DASM allows a router to execute local computations for destination  $j$  as long as  $Z_j^i \cap S_{jP_1}^i \neq \emptyset$ .

If router  $i$  is passive when a message has been received for destination  $j$  or a link-cost change has occurred, and  $D_{jk}^i$ ,  $D_{jk}^i$ ,  $\text{QS}_j^i$ ,  $S_j^i$ ,  $D_{\min}^{ij}$ ,  $Z_j^i$ , and  $\tilde{D}_{jk}^{i*}$  have been updated, then router  $i$  will remain passive (and therefore execute local computations) if  $Z_j^i \cap S_{jP_1}^i \neq \emptyset$ . If this condition is true, then router  $i$  updates several variables:  $\text{RD}_j^i \leftarrow D_{\min}^{ij}$ ,  $S_j^i \leftarrow S_{jP_1}^i$ , and  $\text{FD}_j^i \leftarrow \min(\text{FD}_j^i, D_{\min}^{ij})$ . Router  $i$  then sends updates to all its neighbors if  $\text{RD}_j^i$  has changed.

If  $Z_j^i \cap S_{jP_1}^i = \emptyset$ , then router  $i$  must become active. In this case, if  $S_j^i = \emptyset$ , then  $\text{RD}_j^i \leftarrow \infty$ . Otherwise a value must be chosen. This value can be either  $\infty$  or some value in  $\{D_{jk}^i + l_k^i \mid k \in N^i\}$ , but if  $\text{QS}_j^i \neq \emptyset$ , then the value must be at least as large as  $\min\{D_{jk}^i + l_k^i \mid k \in \text{QS}_j^i\}$ .

### 3.4 Internodal Synchronization

A router becomes active when LRC does not yield any shortest path. At that point, the router must synchronize with its neighbors, *before* it can bring new neighbors to its successor set to reflect the changes caused by the input event that makes the router update its routing table.

Internodal synchronization in DASM is based on generalized diffusing computations, which differ from the diffusing computations used in DUAL in that a router can change its query successor set within a diffusing computation and can reply to the members of its query successor set and become a root of a diffusing computation (DUAL by contrast requires this set contain a single member for the duration of a diffusing computation unless the link to that member fails). Section 4.3 provides conditions under which this can be done.

The key idea is to exploit loop-freedom and to require that a router never empty its successor set unless (a) the router becomes passive with infinite feasible distance or (b) a topology change has made the successor unreachable. With loop-free paths, a path must either reach the destination, or reach a router with an empty successor set. Furthermore, except in response to a link failure, a router  $i$  can prevent a neighboring router from leaving the router  $i$ 's successor set by not replying to a query until router  $i$  will either have some other router in its successor set (this may require that router  $i$  send a query with a sufficiently high feasible distance so another successor can be obtained) or will become passive with infinite feasible distance (in which case all the neighbors will have reported an infinite distance, so a passive router with infinite feasible distance has an empty successor set and the definition of a successor set ensures that such a router is not in any other router's successor set).

The requirement that a router never empty its successor set unless (a) or (b) hold implies that, if router  $i$  has no successors and is also in the successor set of a neighboring router, then router  $i$  must be active and in fact only a link failure can cause router  $i$  to become active while still in the successor set of another node. Hence, as long as an active router eventually becomes passive (conditions that ensure this are given below), then in a stable topology, all routers with no successors will eventually become passive routers that cannot be in the successor set of any other routers. Once this happens, all paths will lead to the destination, and all routers for which the destination is not reachable will have empty successor sets.

We now proceed by describing the internodal synchronization mechanisms in DASM in detail. When a router receives a query from a destination that is not in its successor set, the router must send a reply to ensure that the liveness property given in Section 4.2 holds. Otherwise, when the local computation condition does not hold at router  $i$  for destination  $j$ , router  $i$  begins or joins a generalized diffusing computation by sending queries to its neighbors using the new value chosen for  $\text{RD}_j^i$  in the queries. During this active phase,  $\text{FD}_j^i$  is not increased. At the end of the active phase,  $\text{FD}_j^i$  will increase to  $\text{RD}_j^i$  if router  $i$  starts another active phase, and will assume a value no larger than  $\text{RD}_j^i$  if router  $i$  becomes passive. The behavior of router  $i$  at the end of an active phase (where  $r_{jk}^i = \text{false}$  for all  $k$ ) is determined by a variable  $S_{jP_2}^i$  defined as follows:

$S_{jP_2}^i$ : At the end of an active phase,  $S_{jP_2}^i$  is the successor set that router  $i$  would have for destination  $j$  if router  $i$  becomes passive for destination  $j$ .  $S_{jP_2}^i = \{k \in N^i \mid \tilde{D}_{jk}^{i*} < \min(D_{\min}^{ij}, \text{RD}_j^i)\}$ .

If  $S_{jP_2}^i \cap Z_j^i \neq \emptyset$ , a neighbor in  $S_{jP_2}^i$  will provide the shortest path to destination  $j$ , and router  $i$  can become passive, setting  $S_j^i \leftarrow S_{jP_2}^i$ ,  $\text{RD}_j^i \leftarrow D_{\min}^{ij}$ , and  $\text{FD}_j^i \leftarrow \min(\text{RD}_j^i, D_{\min}^{ij})$ . In becoming passive, router  $i$  will also send replies to all neighbors in  $\text{QS}_j^i$ , and

will send updates as needed to announce changes in  $RD_j^i$ . If  $S_{jP_2}^i \cap Z_j^i = \emptyset$  and  $RD_j^i = \infty$ , then router  $i$  can also become passive. In this case,  $D_{\min}^{ij}$  is also  $\infty$ , and router  $i$  will become passive with  $FD_j^i = \infty$ .

If  $S_{jP_2}^i \cap Z_j^i = \emptyset$  and  $RD_j^i \neq \infty$ , router  $i$  must start a new active phase, and to do this, it must raise its feasible distance. The method used ensures that in a stable topology, router  $i$  can become passive after a finite number of active phases. First, router  $i$  sets  $FD_j^i \leftarrow RD_j^i$ , and then sets  $S_j^i \leftarrow \{k \in N^i \mid D_{jk}^{i*} < FD_j^i\}$ . If  $S_j^i = \emptyset$ , then  $RD_j^i \leftarrow \infty$ ; else if  $QS_j^i \neq \emptyset$ , then  $RD_j^i \leftarrow \min\{D_{jk}^{i*} + l_k^i \mid k \in QS_j^i\}$ , otherwise  $RD_j^i \leftarrow \min\{D_{ijk}^{i*} + l_k^i \mid k \in S_j^i\}$ . Because a neighbor can only raise its feasible distance when the neighbor receives a reply to a query, and because a router does not have to reply immediately to a query from any router in its successor set, a router can keep a neighbor in its successor set long enough for the router to raise its feasible distance sufficiently that the router becomes passive.

Finally, if, during an active phase, an event results in  $Z_j^i \cap S_{jP_1}^i \neq \emptyset \wedge RD_j^i \geq D_{\min}^{ij}$  at router  $i$  for destination  $j$ , then router  $i$  may send replies to all its neighbors in  $QS_j^i$  and updates as required to all other neighbors. For performance reasons, this is done only when  $QS_j^i \cap \{k \in N^i \mid \hat{D}_{jk}^{i*} = \infty\} = \emptyset$ . Under these conditions, router  $i$  would be able to become passive if replies to all its queries had arrived. Upstream neighbors will, of course, behave similarly. Under the appropriate circumstances, the root of the diffusing computation will then move upstream shortening the time needed for convergence.

### 3.5 Handling of Topology Changes

Link failures are treated as implicit replies to outstanding queries sent to a neighbor reachable over the link. Thus, if a failure of link  $(i, k)$  is detected at router  $i$ , router  $i$  sets  $r_{jk}^i \leftarrow \text{false}$  for each destination  $j$ . Because no path may traverse a link that has failed, router  $i$  also sets  $D_{jk}^{i*} \leftarrow \infty$ ,  $\hat{D}_{jk}^{i*} \leftarrow \infty$ , and  $D_{jk}^i \leftarrow \infty$ . DASM then behaves just as it would for any other event.

### 3.6 Loop-Free Area Routing

Suppose each area is represented by a separate entry in the routing tables, and that a destination can belong to more than one area. While one could add routing-table entries to represent such intersections, this would increase table sizes and the length or number of routing-table updates that would have to be sent. We can avoid the need for these additional table entries as follows.

Consider a graph  $G(V, E)$  and let  $J = \{j_1, j_2, \dots, j_m\}$  be a set of  $m$  destinations such that  $J \subset V$ . Let  $FD_J^{i*} = \min\{FD_{j_n}^{i*} \mid j_n \in J\}$ . Furthermore, let  $S_J^{i*} = \{x \mid \forall (j_n \in J) : (D_{jn}^{i*} < FD_J^{i*})\}$ . We define the successor graph  $SG_J$  of graph  $G(V, E)$  for a set of destinations  $J$  at time  $t$  to be a directed graph with the same routers as those in  $G(V, E)$ , but for which a directed edge from router  $i \in V$  to router  $k \in V$  exists if and only if  $k \in S_J^{i*}$ . Loop-free routing follows immediately, as is shown in Theorem 3.

Thus, to route to a router  $v$  that is in more than one area, one can set  $J$  to be the destinations representing the areas that contain  $v$ , and use  $S_J^{i*}$  to determine a successor. No matter what successor in  $S_J^{i*}$  is used, the path will always be loop free at each instant of time.

## 4. Correctness of DASM

The correctness proof for DASM requires proving that (1) DASM is loop free at every instant of time, (2) DASM ensures that a router will not wait indefinitely before receiving replies to queries, (3) in a stable topology DASM converges in  $\bar{C}_j$  (the routers with

no paths to destination  $j$ ), and (4) in a stable topology, DASM converges in  $C_j$  (the routers for which a path to destination  $j$  exists). For purposes of the proof, variables in DASM are modeled as functions of time, with an equivalent definition of  $D_{jk}^{i*}$  that avoids the need to use the variable  $\hat{D}_{jk}^{i*}$ .

### 4.1 Loop Freedom

For a router  $i$ , a destination  $j$ , and neighbor  $k$  of router  $i$ ,  $FD_j^i(t)$  is the feasible distance at router  $i$ ,  $RD_j^i(t)$  is the distance that router  $i$  will report in an update,  $M_{jk}^{i*}$  is the set of times at which router  $i$  processes a message from router  $k$  for destination  $j$ , and  $\text{xmt}_i^k(t)$  is the time at which a message processed by router  $i$  at time  $t$  was sent by router  $k$ .  $D_{jk}^{i*}(t)$  is defined as follows:

*Definition 2:* Let  $G(V, E)$  be a graph, let  $i \in V$  be a router, let  $j \in V$  be a destination, and let  $k \in N^i$  be a neighbor of router  $i$ . Then  $D_{jk}^{i*}(t)$  is a function satisfying the following conditions:

- When link  $(i, k)$  is not operational or has just recovered, or when router  $i$  initializes itself at time  $t$ ,  $D_{jk}^{i*}(t) = \infty$ .
- Suppose router  $i$  sends a reply to router  $k$  at time  $t_r$ . Let  $t_1$  be the last time before or at  $t_r$  at which either a query from router  $k$  was processed, link  $(i, k)$  recovered, or router  $i$  initialized itself. Then  $D_{jk}^{i*}(t_r) = \min\{RD_j^k(\text{xmt}_i^k(\tau)) \mid \tau \in M_{jk}^{i*} \cap [t_1, t_r]\}$ .
- Otherwise,  $D_{jk}^{i*}(t) = \min(\{RD_j^k(\text{xmt}_i^k(\tau)) \mid \tau \in M_{jk}^{i*} \cap [t_2, t]\} \cup \{D_{jk}^{i*}(t_2)\})$ , where  $t_2$  is the last time before  $t$  when router  $i$  either sent router  $k$  a reply, detected a link recovery for link  $(i, k)$ , or initialized itself.

*Lemma 1:* Let  $G(V, E)$  be a graph in which each router runs DASM. If router  $i \in V$  initializes itself or receives the last reply to a query for destination  $j \in V$  at time  $t$ , all undelivered messages (if any) sent by router  $i$  for destination  $j$  will contain a distance greater than or equal to  $RD_j^i(t^-)$ , the value immediately before router  $i$  processes the last reply.

*Proof:* When a router initializes itself, there are no undelivered messages. The lemma holds in this case.

DASM requires that all messages be delivered in FIFO order, and that all messages that are undelivered before a link failure are lost. Thus, all messages that router  $i$  originated before sending a query will either have been lost or delivered. For some destination, the reply distance of a router  $i$  must be a decreasing function of time over the interval between sending a query for that destination and receiving the last reply to that query. Accordingly, all messages from router  $i$  that are undelivered when the last reply to that query is received must contain a value greater than or equal to  $RD_j^i(t^-)$ . ■

*Lemma 2:* Let  $G(V, E)$  be a graph in which each router runs DASM, and let  $j \in V$  be a destination. At time  $t$ , when a router  $i \in V$  initializes itself or receives its last reply if it had become active, any neighbor  $k \in N^i$  that can use router  $i$  as a feasible successor for some destination  $j$  will have either set  $D_{ji}^{k*}(t)$  to some value such that  $RD_j^i(t^-) \leq D_{ji}^{k*}(t)$ .

*Proof:* At  $t = 0$ , all routers initialize themselves and set the distance to their neighbors to  $\infty$ . The lemma is consequently true at time  $t = 0$ .

At any point at which a router has become active, or at which it will stay active after receiving replies from all its neighbors, the router will send queries to all its neighbors when originating a diffusing computation, and will send queries to all its neighbors.

Suppose that, at time  $t_2$ , router  $i$  receives its last reply to a query for destination  $j$  sent at time  $t_1$ . Let  $RD_j^i(t_1^+)$  be the value of the reply distance for router  $i$  that will be sent in the query generated at time  $t_1$ , and let  $RD_j^i(t^-)$  be the value of the reply distance for router  $i$  just before it receives its last reply. Because

$RD_j^i$  is a decreasing function in  $(t_1, t)$ ,  $RD_j^i(t_1^+) \geq RD_j^i(t^-)$ . Because router  $i$  is active for all  $t \in (t_1, t)$ , and according to DASM's operation, router  $i$  can only send messages for destination  $j$  containing a distance  $RD_j^i(\tau)$  at some time  $\tau \in (t_1, t)$  if  $RD_j^i(\tau) \leq \min\{RD_j^i(t') \mid t' \in (t_1, t)\}$ . Suppose that router  $i$  has sent a query to router  $k$ . Unless the link from router  $i$  fails in  $(t_1, t)$ , router  $k$  will have processed that query and router  $i$  will have received its reply sometime at or before  $t$ . When router  $k$  sends its reply at time  $t_r$ , it will have received a query containing the value  $RD_j^i(t_1^+)$  and will have set  $D_{ji}^{k*}(t_r)$  to the minimum value or  $RD_j^i$  received in the query or a subsequent update. Because  $RD_j^i$  is a decreasing function of time in  $(t_1, t)$ , and because router  $k$  received the query transmitted from router  $i$  at time  $t_1$ ,  $RD_j^i(t^-)$  is less than or equal to the value in the last message received by router  $k$  before time  $t$ . Thus,  $RD_j^i(t^-) \leq D_{ji}^{k*}(t)$ .

If the link has failed at some time  $t_f$  during  $(t_1, t_2)$ , router  $k$  will set  $D_{ji}^{k*}(t_f) = \infty$ . If a message from  $i$  is subsequently received during  $(t_f, t)$ , router  $k$  can set  $D_{ji}^{k*}(t)$  to some value such that  $D_{ji}^{k*}(t) \in \{RD_j^i(\tau) \mid \tau \in (t_f, t)\}$ . This is also true if the link fails multiple times. If the link had failed at or before  $t_1$  and recovered (or was initially established) after  $t_1$ , then router  $k$  may or may not have received a message (which must have been generated after  $t_1$ ) from router  $i$  for destination  $j$ . If router  $k$  has not received this message, then  $D_{ji}^{k*}(t) = \infty$  in which case  $RD_j^i(t^-) \leq D_{ji}^{k*}(t)$ , otherwise  $D_{ji}^{k*}(t_2) \in \{RD_j^i(t') \mid t' \in (t_1^+, t_2^-)\}$ , in which case  $RD_j^i(t^-) \leq D_{ji}^{k*}(t)$  because  $RD_j^i$  is a decreasing function of time in  $(t_1, t_2)$ . ■

*Lemma 3:* Let  $G(V, E)$  be a graph in which each router runs DASM. Let  $i$  and  $k$  be routers in  $V$ , and  $j$  be a destination. Furthermore, let  $k \in N^i$ . Suppose that, at time  $t_a$ ,  $FD_j^i(t_a) \leq D_{ji}^{k*}(t_a)$ , all messages that are undelivered (but not lost) at time  $t_a$  and that are generated by router  $i$  before  $t_a$  contain a distance at least as large as  $FD_j^i(t_a)$ , and that  $FD_j^i$  is a decreasing function of  $t$  for  $t \in [t_a, t_b]$ . Then  $FD_j^i(t) \leq D_{ji}^{k*}(t)$  for all  $t \in [t_a, t_b]$ .

*Proof:* If the link from router  $i$  to router  $k$  fails at some time  $t_f \in [t_a, t_b]$ , where  $t_f$  is the earliest time in this interval at which such a link failure occurs, then router  $k$  will set  $D_{ji}^{k*}(t_f)$  to infinity, and reset the value only when a new message for destination  $j$  is received from router  $i$ . The lemma thus holds when  $D_{ji}^{k*}(t_f) = \infty$ . In the interval  $[t_f, t_b]$ , the only messages received by router  $k$  will have been generated by router  $i$  after  $t_f$  because of the operation of the neighbor-to-neighbor protocol. Thus, there are no messages are undelivered at time  $t_f$  that will be received after  $t_f$ , and by assumption,  $F_j^i$  is a decreasing function of  $t$  in  $[t_f, t_b]$ , and all the conditions the lemma requires are therefore true over the interval  $[t_f, t_b]$  if these conditions are true for  $[t_a, t_b]$ . It is therefore sufficient to prove that the lemma is true only in the case where there are no failures of link  $(i, k)$  in the interval  $[t_a, t_b]$ .

By assumption, all messages generated by router  $i$  before  $t_a$  and received by router  $k$  contain distances larger than  $FD_j^i(t_a)$ , and  $FD_j^i(t) \leq FD_j^i(t_a)$ . The lemma therefore holds when any of these messages are delivered to router  $k$ , and if no other messages or no messages at all are delivered to router  $k$  from router  $i$  during  $[t_a, t_b]$ , the lemma is true. The remaining case is the one where at least one message generated after time  $t_a$  is received before time  $t \in [t_a, t_b]$ . For any message generated at time  $t' \in [t_a, t_b]$  for destination  $j$ , DASM requires that  $RD_j^i(t') \geq FD_j^i(t')$ , and because  $FD_j^i$  is a decreasing function of time in  $[t_a, t_b]$ , it follows that  $FD_j^i(t) \leq \min\{RD_j^i(\tau) \mid \tau \in [t_a, t]\}$  for  $t \in [t_a, t_b]$ . Similarly, because  $D_{ji}^{k*}(t_a) \geq FD_j^i(t_a)$ , because all messages generated before  $t_a$  and received after  $t_a$  contain distances larger than  $FD_j^i(t_a)$ , and because the definition of  $D_{ji}^{k*}$  requires it to con-

tain the minimum distance seen in a message after either a router initialized itself (at which point the value is infinity), the link recovered (at which point the value is infinity), or a query was received for which a reply has been sent, it follows that  $D_{ji}^{k*}(t) \geq \min\{FD_j^i(t_a)\} \cup \{RD_j^i(\tau) \mid \tau \in [t_a, t]\}$  for  $t \in [t_a, t_b]$ . Because  $FD_j^i(t) \leq \min\{RD_j^i(\tau) \mid \tau \in [t_a, t]\}$ , it is obviously true that  $FD_j^i(t) \leq \min(\{FD_j^i(t_a)\} \cup \{RD_j^i(\tau) \mid \tau \in [t_a, t]\})$ . Consequently,  $FD_j^i(t) \leq D_{ji}^{k*}(t)$  for  $t \in [t_a, t_b]$ . ■

*Theorem 1:* Let  $G$  be a graph in which each router runs DASM. Let  $i, j$  and  $k$  be routers in  $G$ , let  $j$  be a destination, and let  $k \in N^i$ . Then  $FD_j^i(t) \leq D_{ji}^{k*}(t)$  is true for all  $t \geq 0$ .

*Proof:* Consider the times in the interval  $[0, t]$  at which router  $i$  either initializes itself, loses all its neighbors, or receives the last reply to a query. These times form a sequence  $L = \{t_0 = 0, t_1, t_2, \dots, t_n < t\}$ , where  $t_0$  is the time at which router  $i$  initializes itself, and  $t_1, t_2, \dots, t_n$  are the times at which router  $i$  receives the last reply from some query. DASM requires that the feasible distance of router  $i$  always decrease except at the times in  $L$ . When a router first initializes itself, the theorem is true because all its neighbors will set their distance-table entry for that router to  $\infty$  for destination  $j$  until they receive a message for destination  $j$  containing a lower distance. Other than at time  $t = 0$ , DASM allows a router to increase its feasible distance only when it receives replies from all its neighbors, or when it has lost all neighbors (in which case its neighbors will set their distance for that router to  $\infty$ ). Thus, in the interval  $[0, t_1]$  the Lemma 3 applies and  $FD_j^i(t) \leq D_{ji}^{k*}(t)$  for all  $t \in [0, t_1]$ .

The proof proceeds by induction. Consider any time  $t_m \in L$  and assume the theorem holds up to that time. According to Lemma 1, at any time  $t \in [t_m, t_{m+1}]$ , any undelivered message for destination  $j$  that router  $i$  sent before  $t_m$  will contain a distance greater than or equal to  $RD_j^i(t_m^-)$ . By Lemma 2, for any neighbor  $k$  of router  $i$ ,  $D_{jm}^{k*}(t_m)$  will be larger than  $RD_j^i(t_m^-)$ . When router  $i$  receives its last reply at time  $t_m$ , it will change its feasible distance to  $FD_j^i(t_m) = RD_j^i(t_m^-)$ . Because DASM requires that at any router,  $FD_j^i(t) \leq RD_j^i(t)$ , it follows that  $FD_j^i(t_m) \leq D_{jm}^{k*}(t_m)$ . Because  $FD_j^i(t)$  is a decreasing function in the interval  $[t_m, t_{m+1}]$ , Lemma 3 implies that  $FD_j^i(t) \leq D_{ji}^{k*}(t)$  for all  $t \in [t_m, t_{m+1}]$ . We can therefore proceed by induction up to time  $t$ . ■

*Theorem 2:* Let  $G(V, E)$  be a graph in which each router runs DASM. Let  $P_j(t)$  be a path through  $G$  for destination  $j$  at time  $t$  for which each  $i \in P_j(t)$  (other than the last router) chooses a router  $s$  as the next router in  $P_j(t)$  such that  $s \in S_j^i(t)$ . Then  $P_j(t)$  is loop free.

*Proof:* By definition, each router  $s \in S_j^i(t)$  satisfies  $D_{js}^i(t) < FD_j^i(t)$ . By Theorem 1, for any router  $i \in P_j(t)$ , the next router  $s$  in  $P_j(t)$  will satisfy  $FD_{js}^i(t) \leq D_{js}^i(t)$ . By assumption,  $D_{js}^i(t) < FD_j^i(t)$ ; accordingly, it follows that  $FD_{js}^i(t) < FD_j^i(t)$ . Thus, each subsequent router along the path has a lower value for its feasible distance. If  $P_j(t)$  contained a loop, then for any router  $p \in P_j(t)$  that was part of such a loop, it would follow that  $FD_{jp}^p(t) < FD_j^p(t)$ . Thus, by contradiction, the path must be loop free. ■

The following theorem shows that DASM also provides loop-free area routing:

*Theorem 3:* Let  $G(V, E)$  be a graph in which each router runs DASM. Let  $SG_J(t)$  be the successor graph for graph  $G(V, E)$  and for destination set  $J$  at time  $t$ . Then  $SG_J(t)$  is loop free.

*Proof:* By definition, for each node  $i \in SG_J(t)$ , every neighbor  $k \in N^i \cap SG_J(t)$  satisfies  $k \in S_J^i(t)$ , where  $S_J^i(t) = \{x \mid \forall (j_n \in J) : (D_{jn}^i(t) < FD_J^i(t))\}$ . For  $k \in N^i$ , let

$J_k^*(t) = \{j_n \in J \mid D_{j_n k}^{i*}(t) < FD_j^{i*}(t)\}$ . Clearly, for each  $j_n \in J_k^*(t)$ ,  $D_{j_n k}^i(t) < FD_j^{i*}(t)$ . By Theorem 2,  $FD_{j_n}^k(t) \leq D_{j_n k}^{i*}(t)$ , and thus  $FD_{j_n}^k(t) \leq FD_j^{i*}(t)$ . By the definition of  $FD_j^{i*}(t)$ ,  $FD_{j_n}^{k*}(t) \leq FD_{j_n}^k(t)$ , and therefore  $FD_{j_n}^{k*}(t) < FD_j^{i*}(t)$  for any  $k \in S_j^{i*}(t)$ . Accordingly, if there is a loop  $L_j(t)$  in  $SG_j(t)$  at time  $t$ , then for some  $p \in L_j(t)$ ,  $FD_p^{i*}(t) < FD_j^{i*}(t)$ , and by contradiction,  $SG_j(t)$  is loop free. ■

## 4.2 Liveness

*Theorem 4:* Let  $G(V, E)$  be a graph in which each router runs DASM. If router  $i$  becomes active, it will receive a reply from each neighbor (with link failures generating implicit replies) in a finite time.

The formal proof is essentially the same as the proof for DUAL [5], and can be described informally as follows: The first requirement for showing that generalized diffusing computations terminate is to show that a router that sends a query will always receive a reply to that query and will eventually become passive. To ensure this, several requirements are necessary: a router must reply immediately to a query from any router that is not in its successor set, a router may not raise  $RD_j^i$  during an active phase (excluding its start), and if a new active phase begins immediately after a previous active phase, then the router must raise its reported distance to  $\min(\{D_{j_k}^i(t) + l_k^i(t) \mid k \in QS_j^i(t)\} \cup \{\infty\})$ , and is required to become passive if possible (in which case the router must send replies to any neighbors still in  $QS_j^i(t)$ ). Furthermore, if  $QS_j^i(t) \neq \emptyset$ , router  $i$  should not decrease  $RD_j^i$  during an active phase to a value below  $\min(\{D_{j_k}^i(t) + l_k^i(t) \mid k \in QS_j^i(t)\})$  unless router  $i$  replies to all routers in  $QS_j^i(t)$ .

As a result, router  $i$  will either have sent all the routers in  $QS_j^i$  a reply during an active phase, or will be able to become passive at the end of the active phase, as long as router  $i$  itself receives replies to all the queries it has sent. Thus, a router will reply to a query unless some other router does not reply to its query, which cannot happen: if it did, one can generate a sequence of routers, following the successor graph upstream, until one reach a router that had not replied, but is not in the successor set of any router. Because a router that is not in the successor set of any other router must send replies, it follows by contradiction that all diffusing computations will terminate.

## 4.3 Convergence in $\bar{C}_j$

The proof for convergence in  $\bar{C}_j$  makes use of the following sets:  
*Definition 3:*

$$\begin{aligned} W_{1j}(t) &= \{x \in V \mid x \neq j \wedge S_j^x(t) = \emptyset \wedge \exists y : (y \in V \wedge x \in S_j^y)\} \\ W_{2j}(t) &= \{x \in V \mid x \neq j \wedge S_j^x(t) = \emptyset \wedge \text{active}_j^x(t)\} \\ A_{1j}(t) &= \{x \in V \mid x \neq j \wedge S_j^x(t) \neq \emptyset\} \\ A_{2j}(t) &= \{x \in V \mid x \neq j \wedge S_j^x(t) = \emptyset \wedge \text{passive}_j^x(t)\} \end{aligned}$$

and  $W_j(t) = W_{1j}(t) \cup W_{2j}(t)$ , where  $\text{active}_j^x(t)$  is true if and only if router  $x$  is active for destination  $j$  at time  $t$ , and  $\text{passive}_j^x(t)$  is true if and only if router  $x$  is passive for destination  $j$  at time  $t$ .

These sets are shown in Figure 3. As proven below, DASM ensures that routers “move” between these sets only as shown in Figure 4, and that the set  $W_j(t)$  eventually becomes empty once the topology is stable.  $W_j(t)$  contains all the routers other than the destination at which a path (obtained by following successors) terminates, so once  $W_j(t)$  is empty, all paths must be ones that go to the destination, and all routers that cannot reach the destination must have empty successor sets. ■

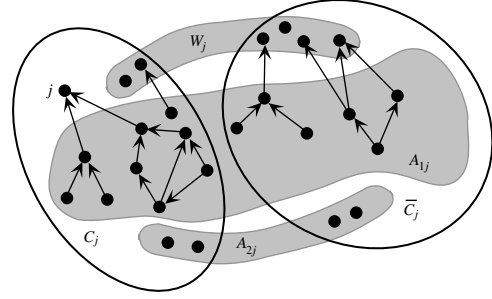


Fig. 3. The Successor Graph and Sets  $W_j$ ,  $A_{1j}$ , and  $A_{2j}$

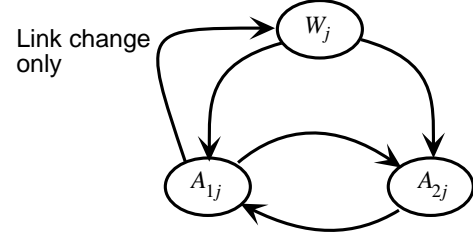


Fig. 4. Transitions between Sets  $W_j$ ,  $A_{1j}$ , and  $A_{2j}$

The operation of DASM is designed to ensure that the following conditions are true:

- R1 If an active phase starts at time  $t$  and  $S_j^i(t) = \emptyset$ , then  $RD_j^i(t) = \infty$ .
- R2 Suppose an active phase at router  $i$  starts at time  $t_q$  and ends at time  $t$ , and that  $RD_j^i(\tau) = \infty$  for all  $\tau \in [t_q, t)$ . Then router  $i$  must become passive at time  $t$ .
- R3 When router  $i$  is passive at time  $t$ , and if  $i \neq j$  and  $S_j^i(t) = \emptyset$ , then  $FD_j^i(t) = \infty$  and  $RD_j^i(t) = \infty$ .
- R4 Suppose router  $i$  is active at time  $t$  and that the active phase at time  $t$  began at time  $t_1 < t$ . Let  $M_j^i$  be the times at which router  $i$  generated a message for destination  $j$ . Then for some  $\tau \in [t_1, t] \cap M_j^i$ ,  $RD_j^i(t) = RD_j^i(\tau)$  where  $RD(\tau) = \infty$  if  $S_j^i(\tau) = \emptyset$ .
- R5 If  $S_j^i(t^-) \neq \emptyset$  and router  $i$  does not process a link event at time  $t$ , then  $S_j^i(t) \neq \emptyset$  unless router  $i$  becomes passive at time  $t$  or stays passive at time  $t$ .
- R6 If a router  $i$  is passive at time  $t^-$  and  $S_j^i(t^-) = \emptyset$ , then router  $i$  must stay passive at time  $t$ .
- R7 if  $FD_j^i(t) = \infty$ , then router  $i$  cannot be in the successor set of any other router.

*Lemma 4:* Let  $G(V, E)$  be a graph for which each router runs DASM and let  $j \in V$  be a destination. Then  $W_j(t) \cup A_{1j}(t) \cup A_{2j}(t) \cup \{j\} = V$ .

*Proof:* Routers are either active or passive, but not both. The lemma follows by direct computation of  $W_j(t) \cup A_{1j}(t) \cup A_{2j}(t) \cup \{j\}$ . ■

*Lemma 5:* Let  $G(V, E)$  be a graph for which each router runs DASM and let  $j \in V$  be a destination. If  $W_j(t) = \emptyset$ , then  $A_{1j}(t) \subset C_j(t)$  and  $\bar{C}_j(t) \subset A_{2j}(t)$ .

*Proof:* If  $A_{1j}(t) \cap \bar{C}_j(t) \neq \emptyset$ , then, because  $SG_j^-(t)$  is acyclic by Theorem 2, starting at some router in  $A_{1j}(t) \cap \bar{C}_j(t)$  and following successors must lead to a router in  $\bar{C}_j(t)$  with no successors. Such a router must be in  $W_{1j}(t)$ , and  $W_{1j}(t) \in W_j(t)$ . Therefore, if  $W_j(t) = \emptyset$ , then  $A_{1j}(t) \cap \bar{C}_j(t) = \emptyset$ . By Lemma 4,  $\bar{C}_j(t) = [W_j(t) \cap \bar{C}_j(t)] \cup [A_{1j}(t) \cap \bar{C}_j(t)] \cup [A_{2j}(t) \cap \bar{C}_j(t)]$ ; therefore, it follows that, if  $W_j(t) = \emptyset$ , then  $\bar{C}_j(t) = A_{2j}(t) \cap \bar{C}_j(t)$ . Accordingly, if  $W_j(t) = \emptyset$ , then  $A_{1j}(t) \subset C_j(t)$  and  $\bar{C}_j(t) \subset A_{2j}(t)$ . ■

*Lemma 6:* Let  $G(V, E)$  be a graph for which each router runs DASM and let  $j \in V$  be a destination. Then all routers in  $W_j(t)$  are active (i.e.,  $W_j(t) = W_{2j}(t)$ ).

*Proof:* Suppose router  $i \in V$  is passive at time  $t$  and that  $i \in W_{1j}(t)$ . By definition of  $W_{1j}$ ,  $S_j^i(t) = \emptyset$  and by R3,  $FD_j^i(t) = \infty$ . Thus, by R7, router  $i$  cannot be in the successor set of any router. Because a router in  $W_{1j}(t)$  must be in the successor set of some other router by Definition 3, the assumption that a passive router is in  $W_{1j}(t)$  is false. By definition, all routers in  $W_{2j}(t)$  are active,  $W_{1j}(t) \subset W_{2j}(t)$ ; therefore,  $W_j(t) = W_{2j}(t)$ . Accordingly, by Definition 3, all routers in  $W_j(t)$  are active. ■

*Lemma 7:* Let  $G(V, E)$  be a graph for which each router runs DASM and let  $j \in V$  be a destination. Then  $A_{1j}(t)$ ,  $A_{2j}(t)$ ,  $W_j(t)$ , and  $\{j\}$  are disjoint sets that cover  $V$ .

*Proof:* By Definition 3,  $\{j\}$  is disjoint from  $A_{1j}(t)$ ,  $A_{2j}(t)$ , and  $W_j(t)$ . Again by Definition 3, both  $W_j(t)$  and  $A_{2j}(t)$  are subsets of  $\{x \in V \mid x \neq j \wedge S_j^x(t) = \emptyset\}$ , and by definition,  $A_{1j}(t) = \{x \in V \mid x \neq j \wedge S_j^x(t) \neq \emptyset\}$ . Clearly,  $A_{1j}(t) \cap W_j(t) = \emptyset$  and  $A_{1j}(t) \cap A_{2j}(t) = \emptyset$ . By definition, all routers in  $A_{2j}(t)$  are passive and by Lemma 6, all routers in  $W_j(t)$  are active. Thus,  $A_{2j}(t) \cap W_j(t) = \emptyset$ . and  $A_{1j}(t)$ ,  $A_{2j}(t)$ ,  $W_j(t)$ , and  $\{j\}$  are consequently disjoint sets. By Lemma 4,  $A_{1j}(t)$ ,  $A_{2j}(t)$ ,  $W_j(t)$ , and  $\{j\}$  cover  $V$ . ■

*Lemma 8:* Let  $G(V, E)$  be a graph for which each router runs DASM, let  $i \in V$  be a router, and let  $j \in V$  be a destination. If  $i \in W_j(t)$ , then at some time  $t' > t$ ,  $i \in A_{1j}(t') \cup A_{2j}(t')$ .

*Proof:* Suppose  $i \in W_j(\tau)$  for all  $\tau \geq t$ . By Lemma 6, router  $i$  is active at time  $t$  and by Theorem 4, must terminate the active phase that exists at time  $t$  at some time  $t_1 > t$ . By assumption,  $i \in W_j(\tau)$ , and according to Lemma 6 all routers in  $W_j(t_1)$  are active; therefore, it follows that router  $i$  must begin another active phase at time  $t_1$ . Because  $S_j^i(t_1) = \emptyset$  if  $i \in W_j(t)$ , then by R1,  $RD_j^i(t_1) = \infty$ . By Theorem 4, this second active phase must terminate at some time  $t_2$ . If  $i \in W_j(\tau)$  for  $\tau \in [t_1, t_2)$ , then  $S_j^i(\tau) = \emptyset$  for  $\tau \in [t_1, t_2)$ , and by R4,  $RD_j^i(\tau) = \infty$  for  $\tau \in [t_1, t_2)$ . Thus,  $\min\{RD_j^i(\tau) \mid \tau \in [t_1, t_2)\} = \infty$ . By R2, router  $i$  must become passive at time  $t_2$ , and hence,  $i \notin W_j(t_2)$ , contradicting the assumption that  $i \in W_j(\tau)$  for all  $\tau \geq t$ . Given that  $j \notin W_j(t)$ , then  $i \neq j$ . Thus by Lemma 7, at some time  $t' > t$ ,  $i \in A_{1j}(t') \cup A_{2j}(t')$ . ■

*Lemma 9:* Let  $G(V, E)$  be a graph for which each router runs DASM, let  $i \in V$  be a router, and let  $j \in V$  be a destination. Suppose that no link events occur after time  $t_c$ . If  $t > t_c$ ,  $i \in A_{1j}(t^-)$ , then  $i \notin W_j(t)$ .

*Proof:* By Lemma 7,  $i \notin W_j(t^-)$  and by Definition 3,  $S_j^i(t^-) \neq \emptyset$ . By Lemma 6, if  $i \in W_j(t)$ , router  $i$  must either be active at time  $t^-$  or change from passive to active at time  $t$ . Because there are no link events after time  $t_c$ , by R5,  $S_j^i(t) \neq \emptyset$  for router  $i$  to be active at time  $t$ . Hence  $i \notin W_j(t)$ . ■

*Lemma 10:* Let  $G(V, E)$  be a graph for which each router runs DASM, let  $i \in V$  be a router, and let  $j \in V$  be a destination. Suppose that no link events occur after time  $t_c$ . If  $i \in A_{2j}(t^-)$  for  $t^- > t_c$ , then  $i \notin W_j(t)$ .

*Proof:* By Definition 3, router  $i$  is passive at time  $t^-$  and  $S_j^i(t^-) \neq \emptyset$ . By R6, router  $i$  must remain passive at time  $t$ , and therefore  $i \notin W_j(t)$  because according to Lemma 6, all routers in  $W_j(t)$  are active. ■

*Theorem 5:* Let  $G(V, E)$  be a graph for which each router runs DASM, and let  $j \in V$  be a destination. If no link events occur after time  $t_c$ , then there must exist a time  $t_f > t_c$  such that for all  $t \geq t_f$ ,  $W_j(t) = \emptyset$  and  $\bar{C}_j(t) \subset A_{2j}(t)$ .

*Proof:* By Lemmas 7 and 8, any router  $i \in W_j(t_c)$  must satisfy  $i \notin W_j(t')$  and  $i \in A_{1j}(t') \cup A_{2j}(t')$  for some time  $t' > t_c$ . By Lemmas 9, 10, the assumption that there are no link events

after  $t_c$ , and the condition  $i \in A_{1j}(t') \cup A_{2j}(t')$ , it follows that  $i \notin W_j(t)$  for all  $t > t'$ . Because  $W_j(t_c)$  can contain only a finite number of routers, there must exist a time  $t_f$  such that  $W_j(t) = \emptyset$  for all  $t \geq t_f$ . Because  $W_j(t) = \emptyset$  for all  $t \geq t_f$ , Lemma 8 implies that  $\bar{C}_j(t) \subset A_{2j}(t)$  for  $t \geq t_f$ . ■

Theorem 5 shows that eventually for destination  $j$ , given a stable network, every router in  $\bar{C}_j$  will be in the set  $A_{2j}(t)$  for  $t > t_f$ , and the definition of  $A_{2j}(t)$  requires that members of this set be passive with an empty successor set. The operation of DASM also ensures that routers in this state have their distance to  $j$  set to infinity, thus ensuring convergence in the  $\bar{C}_j$  with correct distances. Termination follows because DASM does not send messages (other than replies to queries) when passive unless distances change.

#### 4.4 Convergence in $C_j$

The proof for convergence in  $C_j$  is omitted; it is similar to the proof for DUAL [5] and follows by an inductive argument.

The idea behind the proof is to show that, with a stable topology, all routers in  $C_j$  eventually have obtained distances from each neighbor at least as large as the distances along the shortest paths through those neighbors. The proof then shows that the router whose shortest-path distance to the destination is smallest (this must be a neighbor of the destination) will become passive with the correct distance, and only send replies to queries.

### 5. Complexity of DASM

DASM's complexity is measured by the number of steps, called *time complexity* or TC, and the number of messages, called *communication complexity* or CC, required by the algorithms after a single change in the cost or status of a link. To calculate DASM's complexity, it is necessary to assume that the algorithms under study behave synchronously, so that every router in the network executes a step of the algorithm simultaneously at fixed points in time. At each step, a router receives and processes all input events originated during the preceding step, and, if needed, creates an update message after processing each input event; all these update messages are transmitted in the same step. The first step occurs when at least one router detects a topological change and issues update messages to its neighbors. During the last step, at least one router receives and processes updates from its neighbors, after which all routing tables are correct and routers stop transmitting updates until a new topological change takes place.

DASM has the same time and communication complexity as DUAL. After a single link failure or link-cost increase, DASM has  $TC = O(x)$  [5] [10], where  $x$  is the number of routers affected by the routing-table perturbation. To verify this, we note that in the worst case all routers upstream of a destination router  $j$  in  $SG_j$  must participate in a diffusing computation for router  $j$ , which corresponds to the operation of DUAL. DASM's communication complexity is the same as DUAL's, i.e.,  $CC = O(6D \cdot x)$ , where  $D$  is the maximum degree of a router; this is verified in [5]. After a single link addition or link-cost reduction, DASM has  $TC = O(d)$  and  $CC = O(E)$ , where  $E$  is the number of links in the network and  $d$  is the network diameter (i.e., the length of the longest shortest path in hops between any two network routers). To verify this, note that any router that receives a query reporting a distance decrease must be able to find a feasible successor; accordingly, a router that sends a query after its distance decreases must receive immediate replies from all its neighbors, without those neighbors having to forward the query. On the other hand, routers as far as  $d$  hops from the router detecting the change may require to send an update after receiving an update from a neighbor on their successor set, and all routers may have to send an update as a result of the update they

receive from at least one neighbor in their successor sets.

## 6. Average Performance and Consistency Tests

A series of simulations were run using an existing network simulator [17] to compare the performance of DASM with that of DUAL and an ideal link-state algorithm (ILS). The simulator measured the duration (the running time to convergence, with packets propagating between neighbors in unit time), the number of messages processed (the number of updates, queries, and replies summed over all routers in the network) and the number of packets transferred (summed over all links in the network, with each packet containing one or more messages). The ratio of the packet count to message count gives an estimate of the average packet length. The simulator also recorded an operation count that is an approximate measure of the CPU resources needed to run the algorithm, again summed over all routers in the network. For DASM and DUAL, the operation count was incremented once for each message and once for each iteration in a loop iterating over destinations. For ILS, each packet processed results in  $\log_2 V + E$  operations (obtained from the complexity for the best implementations of Dijkstra's shortest-path algorithm).

The simulation assumes that each router responds to a message instantaneously (i.e., that the time required to complete locally performed computations is a small fraction of that needed to send a packet to a neighbor), and consequently the assumption that messages propagate between neighbors in unit time implies that one duration time unit is equivalent to one step in the time-complexity estimate given in Section 5.

To validate the simulation, we ran a series of tests using an ARPANET topology, modeling single-point failures and recoveries for all links and routers. An ARPANET topology was used for this purpose because previous experience has shown that tests performed on topologies with only a few routers are not adequate for detecting implementation errors. After each topology change, the algorithm was allowed to convergence. We also ran several hundred sequences of topology changes, in which each sequence consisted of a random sequence of link-cost changes, link failures, and link recoveries. After each sequence of changes, the algorithm was allowed to run until it converged. In both cases, we compared the shortest-path distances computed by DUAL and DASM (using a previously validated simulation of DUAL [17], and showed that these distances were identical. Every router was used as a destination in these runs to increase the coverage of the tests. Finally, after each step in the simulation, the DASM simulation verified that for destination  $j$  at router  $i$ ,  $QS_j^i \subset S_j^i$ ,  $FD_j^k \leq D_{jk}^{i*}$ ,  $FD_j^k < FD_j^i$  when  $k \in S_j^i$ , that  $k \in S_j^i$  and  $k \in S_j^k$  are not true simultaneously, and that  $FD_j^i \leq RD_j^i$ . In addition, the simulation ran a loop-detection algorithm after each step using only the next-hop information (not the feasible distances) as an independent test that DASM is in fact loop free.

After validation, a series of simulations were run to compare the performance of DASM to that of other algorithms. These simulations used the ARPANET topology that was used for validation. This topology was chosen both for convenience (input files for this topology were available) and size—it is large enough that one cannot predict the outcome of a simulation by reasoning about the step-by-step behavior of an algorithm, but it is also small enough to allow the simulations to run quickly.

We first determined the average performance of the algorithms, measuring duration, message counts, packet counts, and operation counts, after all single-point router and link failures and recoveries, and for random changes in link cost. In all these cases, the performance of DUAL and DASM were nearly identical—differences

Link Cost Increases			
Statistic	DUAL	DASM	ILS
Duration	10.0092	8.708	7.8132
Packet Count	141.949	108.087	166.777
Message Count	505.602	415.319	168.777
Operation Count	597.602	507.319	27395.3

TABLE I  
COMPARISON OF ALGORITHMS FOR LINK-COST INCREASES

were no more than a few percent in any of the performance measures. This is to be expected—the mechanisms that DASM uses to reduce the number of diffusing computations do not provide benefits in these particular cases.

We also simulated a case in which initially all links in the ARPANET topology had unit costs, a link was then chosen at random, and the link's cost was increased by 1.5 units, after which the simulation was allowed to run until it converged. This was repeated for a total of 500 topology changes. Although the fractional change in link costs effects the ability of both DUAL and DASM to avoid starting a diffusing computation, because the link-cost increase was constant, various fractional changes were averaged as a result of the increases in link costs during the course of the simulation—with 500 changes in a 68-link topology, links had their costs increased multiple times. Situations do occur in which costs increase steadily, such as the start of a work day, and thus represent an interesting case. In addition, five separate simulations were run (using a different set of random numbers to select the sequence of link changes), so that statistical errors could be measured. Table I shows the average performance obtained in combining these runs. The largest improvement was in the number of packets transmitted—DUAL requires about 40% more packets than DASM. The variance were lower for DASM than for DUAL, so that cases in which DASM runs significantly longer than the average occur less frequently than with DUAL. The statistical errors in these simulations results are less than 2% for mean values and less than 10% for standard deviations, so the differences seen in Table I are statistically significant. In addition, Table I includes the corresponding numbers for an ideal link state algorithm (labeled ILS), which behaves like OSPF within a single area or within a backbone. The results show a substantial improvement in both CPU utilization and the number of packets transmitted for DASM over ILS (and by inference, OSPF).

A final set of simulations used a model in which link costs changed in fixed increments, varying randomly up and down with a minimum value of 1.0, and in which the simulation is allowed to converge after each link-cost change. We used the Markov chain shown in Figure 5 to model link-cost changes. State 1 in the chain represents a link cost of 1, a state  $i$  other than 1 represents a link cost of  $1 + (i - 1)\delta$ , where the value of  $\delta$  is a constant specified for each simulation. In this chain,  $p$  is the probability that a link cost increases per step and  $q = 1 - p$ . In states other than state 1,  $q$  is the probability of a link cost decreases per step. For the steady-state probabilities, one obtains [11]  $P_1 = (1 - p/q)$  and  $P_{n+1} = \frac{p}{q} P_n$ . This was used to provide an initial set of link costs for a simulation. Subsequently, the simulation can be run long enough so that each link on the average changes its cost 5 times before data collection started, to allow the routing tables to reach typical values (because with DASM, the values of  $S_j^i$  and  $D_{jk}^{i*}$  are dependent on the sequence of previous events, even after convergence).

In Table II, the results of several runs are shown. Both the probability of a link-cost increase and the size  $\delta$  of the increase are varied. These tables suggest that DASM sends fewer messages than DUAL and ILS when link costs vary continuously. In com-

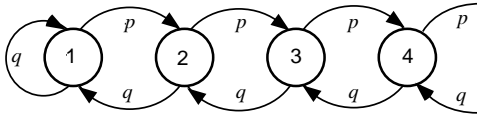


Fig. 5. Markov Chain for Slowly Varying Link-Cost Changes

Link Cost Changes ( $\delta = 1.5, p = 0.45$ )			
Statistic	DUAL	DASM	ILS
Duration	9.6	9.1	7.8
Packet Count	127.2	113.4	166.2
Message Count	527.8	474.0	168.3
Operation Count	619.9	566.0	27299
Link Cost Changes ( $\delta = 1.0, p = 0.45$ )			
Statistic	DUAL	DASM	ILS
Duration	9.3	8.5	7.8
Packet Count	116.8	97.4	166.6
Message Count	480.2	414.0	168.6
Operation Count	572.2	506.0	27370
Link Cost Changes ( $\delta = 0.5, p = 0.45$ )			
Statistic	DUAL	DASM	ILS
Duration	7.8	7.6	7.8
Packet Count	101.6	92.9	167.0
Message Count	402.8	382.40	169.0
Operation Count	494.8	474.4	27275
Link Cost Changes ( $\delta = 1.5, p = 0.40$ )			
Statistic	DUAL	DASM	ILS
Duration	10.24	9.60	7.8
Packet Count	136.2	119.9	166.5
Message Count	564.0	496.3	168.5
Operation Count	656.0	588.3	27158
Link Cost Changes ( $\delta = 1.0, p = 0.40$ )			
Statistic	DUAL	DASM	ILS
Duration	9.49	8.40	7.8
Packet Count	119.0	98.1	166.5
Message Count	476.9	402.9	168.5
Operation Count	568.98	494.9	27346
Link Cost Changes ( $\delta = 0.5, p = 0.40$ )			
Statistic	DUAL	DASM	ILS
Duration	7.9	7.7	7.8
Packet Count	102.3	93.6	166.8
Message Count	397.7	376.6	168.9
Operation Count	489.7	468.6	27409

TABLE II  
RANDOM LINK-COST CHANGE

parison to DUAL, the improvement is larger the higher the value of  $\delta$  because with low values of  $\delta$ , diffusing computations occur less frequently. As  $\delta$  is decreased, however, both algorithms do significantly better than ILS (and hence OSPF when area routing is used). Based on the duration, it would appear that ILS converges slightly faster. The additional time that DUAL and DASM take, however, involves cleanup messages related to diffusing computations. DASM uses slightly less computational resources than DUAL for these cases, and both use about 50 times less than ILS.

## 7. Conclusions

DASM provides enhanced functionality over all prior routing algorithms by providing loop-free multipath routing and support for loop-free area routing. With DASM, routers can use multiple loop-free paths of different lengths at any time; if routers are in the intersection of two areas, the intersection does not require an additional

entry in the routing tables.

DASM provides similar performance to DUAL for link or router failures and recoveries, and provides better performance in some cases. The evidence indicates that DASM performs significantly better than DUAL and ILS for the case where link-costs changes are incremental, which would be the case in practice for networks and internets in which link costs are related to congestion—e.g., by basing the cost of links on moving averages of various quantities that can be readily measured, such as queue length or delay. DASM thus appears to offer both better performance and functionality over link-state algorithms and other algorithms based on diffusing computations.

## References

- [1] L. Bosack, "Method and Apparatus for Routing Communications among Computer Networks," U.S. Patent 5,088,032, Cisco Systems, Menlo Park, California, February 1992.
- [2] J. Cain, S. Adams, M. Noakes, and E. Althouse, "A Near-Optimum Multiple Path Routing Algorithm for Space-Based SDI Networks," *Proc. IEEE MILCOM '87*, Washington, D.C., pp. 29.3.1-29.3.7, 1987.
- [3] J. Cain, S.L. Adams, and M. Noakes, "Multiple-Path Routing Mechanism for Packet Communications Network," U.S. Patent No. 4,905,233, Harris Corp., Melbourne, Florida, February 1990.
- [4] E.W. Dijkstra and C.S. Scholten, "Termination Detection for Diffusing Computations," *Information Processing Letters*, Vol. 11, No. 1, August 1980, pp. 1-4.
- [5] J.J. Garcia-Luna-Aceves, "Loop-Free Routing Using Diffusing Computations," *IEEE Trans. Networking*, Vol. 1, No. 1, 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*, Vol. 13, No. 8, 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] M.L. Gardner, I.S. Loobeeck, and S.N. Cohn, "Type-of-Service Routing: Preliminary Design," BBN Report No. 6195, BBN Communications Corporation, 1986.
- [9] P. Humblet, "Another Adaptive Distributed Shortest Path Algorithm," *IEEE Trans. Commun.*, Vol. COM-39, No. 6, 1991, pp. 995-1003.
- [10] J.M. Jaffe and F.M. Moss, "A Responsive Routing Algorithm for Computer Networks," *IEEE Trans. Commun.*, Vol. COM-30, No. 7, July 1982, pp. 1758-1762.
- [11] L. Kleinrock, "Queueing Systems," *John Wiley & Sons*, 1975.
- [12] Y. Rekhter and T. Li, "A Border Gateway Protocol 4 (BGP-4)," RFC 1654, July 1994.
- [13] N. Lynch, *Distributed Algorithms*, Morgan-Kaufmann, 1996, Ch. 9.
- [14] P.M. Merlin and A. Segall, "A Failsafe Distributed Routing Protocol," *IEEE Trans. Commun.*, Vol. COM-27, No. 9, September 1979, pp. 1280-1288.
- [15] J. Moy, "OSPF Version 2," RFC 1583, March 1994.
- [16] A. Segall, "Distributed Network Protocols," *IEEE Trans. Inform. Theory*, Vol. IT-29, No. 1, January 1983, pp. 23-35.
- [17] W. Zaumen and J.J. Garcia-Luna Aceves, "Dynamics of Link-State and Loop-Free Distance-Vector Routing Algorithms," *Journal of Internetworking*, Vol. 3, 1992, pp. 161-188.