Loop-Free Constrained Path Computation for Hop-By-Hop QoS Routing ⋆

Zhenjiang Li [a,*] J.J. Garcia-Luna-Aceves [a,b]

[a] *Department of Computer Engineering, University of California, Santa Cruz*
1156 high street, Santa Cruz, CA 95064, U.S.A.
Phone: 1-831-4595436, Fax: 1-831-4594829

[b] *Palo Alto Research Center (PARC)*
3333 Coyote Hill Road, Palo Alto, CA 94304, U.S.A.

Abstract

We present the distributed multi-constrained routing (DMR) protocol for the computation of constrained paths for QoS routing in computer networks. DMR exploits distance vectors to construct a logical shortest multipath (LSM) for each destination with regard to a given optimization metric, from which a set of non-dominated paths are locally derived at each node. As such DMR is able to find feasible paths as well as optimize the utilization of routing resources.

DMR operates in line with the hop-by-hop, connectionless routing model assumed in the IP Internet, and maintains instantaneous loop freedom. Nodes running DMR need not maintain a global view of network state (topology and resource information), and routing updates are sent only to neighboring nodes. This is in sharp contrast with all previous approaches that rely on complete or partial network state for constrained path computation, which incurs excessive communication overhead in large networks and is hard to achieve in practice.

Key words: QoS routing, constrained path, logical distance, loop freedom

1 Introduction

The goal of quality-of-service (QoS) routing is two-fold: (a) finding the *feasible paths* from source to destination that satisfy multiple constraints simultaneously (e.g., bandwidth, reliability, end-to-end delay and jitter); and (b) routing the traffic in such a way that network resources

⋆ This work was supported in part by the Baskin Chair of Computer Engineering at UCSC, the National Science Foundation under Grant CNS-0435522, and the U.S. Army Research Office under grant No. W911NF-05-1-0246.

* Corresponding author
Email addresses:
zhjli@soe.ucsc.edu (Zhenjiang Li),
jj@soe.ucsc.edu (J.J. Garcia-Luna-Aceves).

are efficiently utilized. The first is the multi-constrained path (MCP) computation problem that is known to be NP-complete, and the second is called the multi-constrained path optimization (MCPO) [1].

As Section 2 describes, existing distributed QoS routing approaches have one or more of the following limitations: (1) requiring the availability of timely global network state at each node, and some approaches even assume that the distribution of routing constraints is known [2], which is hardly achievable in practice due to the dynamic nature of QoS-related parameters (e.g., residual bandwidth, queue lengths) and non-negligible propagation delay of communication mediums; (2) addressing QoS routing subject to only a single constraint (e.g., bandwidth [3,4]); (3) computing only the shortest paths with regard to (w.r.t.) an optimization metric, and as such may not satisfy multiple constraints simultaneously [2,3,5,6,20]; and (4) considering only either multi-constrained path computation or path optimization, but not both, even though they are strongly related with each other.

Hence, a solution is needed for distributed multi-constrained routing that (a) adheres to the IP routing model (hop-by-hop and connectionless oriented), (b) does not require global network state to be made available at each node, and (c) finds multi-constrained paths while optimizing the overall routing performance according to the given optimization metric. This is the subject of this paper.

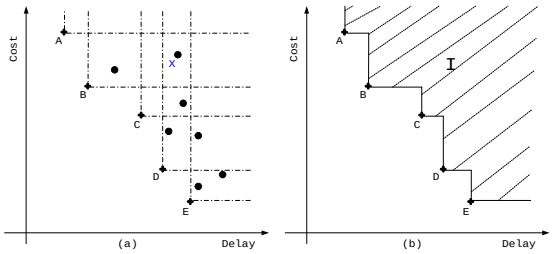

Fig. 1. Non-dominated paths in $\{Cost \times Delay\}$ space

The path measurement of a minimal metric (e.g., bandwidth) is determined by the minimal value of this metric of all links in the path; while that of an additive (e.g., delay) or multiplicative metric (e.g., loss possibility) equals to the sum or product of its values of links along the path. Multiplicative metrics are not considered because they can be translated into additive metrics by using logarithm.

Mieghem and Kuipers [9], and Smith and Garcia-Luna-Aceves [6] have shown that maintaining only non-dominated paths for each destination is sufficient to compute constrained feasible paths. Path p is *dominated* by path q if

$$\begin{cases} w_i^q \leq w_i^p, \text{ for additive metrics} \\ w_i^q \geq w_i^p, \text{ for minimal metrics} \end{cases} \quad (1)$$

for all $i = 1, 2...k$, if there are k routing metrics being considered. A path is called *non-dominated* if it is not dominated by any other path. Using path domination reduces the complexity of constrained path computation, because the capability of QoS provisioning from source to destination can be accurately represented by the set of non-dominated paths be-

tween them. Figure 1 depicts a set of points in $\{Cost \times Delay\}$, which correspond to the set of paths between a source-destination pair, when the link weight consists of cost and delay. In Figure 1 (a), the path represented by point X is dominated by B, which has shorter delay and less cost. However, there exists no clear *better or worse* relation between any two paths amongst $\{A, B, C, D, E\}$, because none dominates another, and any routing request falls in the *feasible area* (shadowed area I of Figure 1 (b)) can be supported by at least one of such non-dominated paths.

We present the distributed multi-constrained routing (DMR) protocol, which constitutes the first solution for the distributed computation of non-dominated loop-free paths subject to multiple constraints that does not require each node to maintain complete network state. DMR consists of three main components: (a) ordering of routes based on logical distances, which are optimization metrics obtained from the various weight components of links (e.g., bandwidth, delay, reliability, etc.); (b) establishing a logical shortest multipath (LSM) for each destination w.r.t. the specified optimization metric; and (c) tracking the raw weight of every path reported by neighboring nodes when establishing LSMs to derive a set of non-dominated paths[1] for each destination.

[1] Extensive simulations show that satisfying success ratios can be achieved by maintaining only a small number of non-dominated paths, e.g., 5.

Section 2 summarizes related work and Sec. 3 describes the network model used in our discussion. Section 4 describes DMR in detail. Section 5 presents its complexity analysis. Section 6 compares DMR with other QoS routing algorithms, and shows that DMR attains similar performance but without requiring each node to have complete network state as in all prior solutions.

2 Related Work

Path selection subject to two constraints is not strong NP-complete, and the computation complexity depends on the values of link weight in addition to the network size [10]. Accordingly, Chen and Nahrstedt [11] proposed to scale one component of the link weight down to an integer that is less than $\lceil \frac{w_i \cdot x}{c_i} \rceil$, where x is a pre-defined integer and c_i is the corresponding constraint on weight component w_i. They prove that the problem after weight scaling is polynomially solvable by an extended version of Dijkstra's (or Bellman-Ford) algorithm, and any solution to the latter is also a solution to the original MCP. The running time of the extended Dijkstra's and Bellman-Ford algorithms are $O(x^2N^2)$ and $O(xEN)$, respectively, where E is the number of links and N is the number of nodes. As we have shown [7], satisfactory performance is achievable only when parameter x is large enough (e.g., $x = 10$), which incurs considerable computation complexity at the source nodes.

Another commonly used scheme for MCP is to define a good link-cost (or path-weight) aggregation function based on the routing metrics being considered. Then any shortest path algorithm can be used to compute the optimal path w.r.t. the resulting aggregated metric. Jaffe [10] was the first to use a linear link-cost function $w(u,v) = \alpha w_1(u,v) + \beta w_2(u,v)$, in which $\alpha, \beta \in Z^+$. Ever since, both linear and non-linear aggregation functions have been proposed. The major limitation of this approach is that the ability to find feasible paths based on an aggregated metric largely depends on the *quality* of the function being used, and most of them are empirical heuristics. Consequently, the shortest path computed w.r.t. a singular metric may not simultaneously satisfy the multiple constraints being considered.

The implementation strategies for QoS routing can be classified into centralized source routing and distributed routing. Most constrained routing algorithms proposed to date use centralized approaches, i.e., they compute feasible paths at each source, and simply assume the availability of global network state whenever they are needed. Centralized MCP algorithms suffer from high computation complexity at source nodes, sluggish response to network changes, and excessive overhead required by the dissemination of topology and resource information throughout the network, which significantly limits their scalability.

Distributed MCP algorithms compute feasible paths locally at each node, and forward packets based only on their destination addresses on a hop-by-hop basis. Distributed routing is more responsive, robust and scalable than centralized schemes, because nodes independently make routing decision and therefore are able to respond to network dynamics quickly.

Based on TAMCRA [13], and its exact modification SAMCRA, Mieghem et al proposed a hop-by-hop destination-based only (HbHDBO) QoS routing protocol [2], which has a worst-case complexity $O(kNlog(kN) + k^2CE)$, where k is the number of non-dominated paths maintained at each node and C is the number of constraints being considered. A node that runs HbHDBO uses a modified Dijkstra's algorithm to compute k non-dominated paths for each destination, based on a non-linear weight function $w(p) = max\left(\frac{w_i(p)}{c_i}\right)$, where c_i is the constraint on metric w_i. For HbHDBO to work correctly, global network state is required at each node, and routing constraints must also be known a priori. However, in the worst case, k can grow exponentially large, and the performance of HbHDBO also varies with the number of constraints.

Sobrinho adopts an algebraic approach and investigates the path optimization problem in the context of Hop-by-Hop QoS routing [5]. In Sobrinho's algebraic framework, routing is separated into path weight functions that define routing optimization requirements (e.g., widest-shortest path, most-reliable path), and the algorithms that compute the

optimal paths based on the aggregated metric defined by the weight function being used. Sobrinho establishes the algebraic properties that a path weight function must have for any routing algorithm to converge correctly (i.e., to give the optimal paths w.r.t. the aggregated metric). Though the results obtained by Sobrinho establish a generalized framework for QoS-oriented path optimization, they cannot be applied to constrained path optimization, because paths are optimized only w.r.t. the given path weight function, rather than being computed to satisfy multiple constraints.

On the basis of Sobrinho's work, Smith and Garcia-Luna-Aceves [6] proposed an improved algebra to address multi-constrained path computation, in which a new partial order relation $\sqsubseteq$ is introduced. $\sqsubseteq$ is used to select the *maximal element* from a set of comparable paths, such that only those non-dominated paths are kept for each destination. A boolean traffic algebra is also introduced to describe the traffic allowed over a link, such that policy oriented constraints (for traffic engineering) can also be expressed and considered in the path computation. Ignoring the cost spent on the evaluation of boolean expressions, the proposed generalized Dijkstra's algorithm has a complexity of $O(NW^2A^2)$, where W is the maximal value of link weights and A is the maximal number of true assignments in the traffic algebra. The major limitation of approaches based on a traffic algebra is that *boolean satisfiability* itself is NP-Complete, which unfortunately is not taken into account for the performance analysis of the proposed algorithms.

3 System Model

We model the network as a directed graph $G = \{V, L\}$, where V is the set of nodes and L is the set of links interconnecting the nodes. N and E are the cardinalities of V and L, i.e., $N = |V|$ and $E = |L|$, respectively.

We assume that each link $l_{u,v}$ is associated with a link weight vector $w(u,v) = \{w_1, w_2 ... w_k\}$, in which w_i is an individual weight component, i.e., a single routing metric. Accordingly, any path p from a source to a destination can be assigned a path weight vector $w^p = \{w_1^p, w_2^p ... w_k^p\}$, where $w_i^p = \sum_{l_{u,v} \in p} w(u,v)$, if w_i is an additive metric (e.g., delay); or $w_i^p = min(w_i(u,v)),\ l_{u,v} \in p$, if w_i is a minimal metric (e.g., bandwidth).

The *logical distance (LD)* of path p is given by a path function f^p based on the weights of its consisting links. In QoS routing, f^p is usually used to specify how the routing should be optimized. For example, in bandwidth-inversion shortest-path routing, $f^p = \sum_{l_{u,v}} \frac{1}{B(u,v)}, l_{u,v} \in p$, where $B(u,v)$ is the available bandwidth of $l_{u,v}$, such that the shortest path, in term of inverted bandwidth, is preferred. In contrast, in most-reliable path routing, we can define $f^p = \sum_{l_{u,v}} -\log\left(prob(u,v)\right), l_{u,v} \in p$, where $prob(u,v)$ is the reliable probability of $l_{u,v}$, such that the most reliable path is preferred. We call f^p the *optimization function*, and the

logical distance computed by f^p the *optimization metric*, given that the path with minimal logical distance is also optimal w.r.t. the optimization requirements implied by f^p. Note that an optimization metric should not be confused with an actual *routing metric*, which can be bandwidth or delay for instance.

It is worth pointing out that logical distances are **not** necessarily simple real numbers. A logical distance is a real number only if the optimization function f^p can be given by a close-form expression. This is the case for the bandwidth-inversion shortest path ($\sum_{l_{u,v}} \frac{1}{B(u,v)}$), the most-reliable path ($\sum_{l_{u,v}} -\log(prob(u,v))$), or the aggregated metric used in the Interior Gateway Routing Protocol (IGRP) [15]: $f^p = L + \frac{k}{C}$, where k is a positive constant, L and D are the path length and capacity, respectively. Otherwise, a logical distance can be a tuple consisting of multiple routing metrics. For example for widest shortest-path (WSP) routing, a path with the shortest distance (e.g., hops) is preferred, and if multiple such shortest paths exist, the one with the maximal bandwidth is preferred. The f^p for WSP is then defined by $(\sum D_{u,v}, min(B_{u,v}))$, in which $D_{u,v}$ and $B_{u,v}$ are the distance and bandwidth of each comprising link $l_{u,v}$ along the path. As a result, to compare the precedence between two paths, only bandwidth or distance is not enough. Tuple $< D, B >$ has to be used as the logical distance for each path (D and B are distance and bandwidth, respectively), and path p with logical distance $<D1, B1>$ is better than path q with $<D2, B2>$, only if

$$D1 < D2 \vee (D1 = D2 \wedge B1 > B2) \tag{2}$$

Similarly, in least-cost shortest-path routing (LCSP), tuple $< D, C >$ is used as the logical distance for each path, in which D is the distance and C is the cost. Function f^p is defined by $(\sum D_{u,v}, \sum C_{u,v})$ over all comprising link $l_{u,v}$ of a path. Path p with $< D1, C1 >$ is preferred over path q with $<D2, C2>$, only if

$$D1 < D2 \vee (D1 = D2 \wedge C1 < C2) \tag{3}$$

By extrapolating the real-number metrics used in conventional best-effort routing to logical distances, it is handy to compute optimal paths in the context of QoS routing, provided that a total order properly exists amongst the logical distances defined by the optimization function f^p being used.

4 Distributed Multi-Constrained Routing (DMR) Protocol

4.1 Principles of Operation

Let LD^i_j denote the logical distance from node i to destination j as known by node i. LD^i_{jk} denotes the logical distance LD^k_j from node k, which is a neighbor of node i, to destination j, as reported to node i by node k. FLD^i_j denotes the *feasible logical distance (FLD)* of node i for destination j, which is an estimate of the minimal logical distance maintained for destination j by node i.

A node i that runs DMR maintains a routing entry for each destination j, which includes FLD^i_j, LD^i_j and the successor set chosen for j (denoted by S^i_j) . Node i maintains a neighbor table that records the logical distance LD^i_{jk} reported by each node k in its neighbor set N^i for each destination j; and a link table that reflects the link state $w(i,k)$ for each adjacent link $l_{i,k}, k \in N^i$.

Given that each node must run DMR for each destination, we focus on the operation of node i's computation of the set of non-dominated paths for a destination j. Provided that each node maintains up to x non-dominated path for destination j, node i may receive and record x values of LD^i_{jk} from each neighbor k; node i also reports to its neighbors the logical distances of the x non-dominated paths from itself to destination j, of which the minimal value is also used as the feasible logical distance FLD^i_j of node i. For destination j we have $LD^j_j = \bar{0}$, $FLD^j_j = \bar{0}$, and $LD^k_{jj} = \bar{0}, \forall k \in N^j$, where $\bar{0}$ is the *zero* as defined by f^p. When a node is powered up, FLD is set to $\bar{\infty}$, the *infinity* as defined by f^p, and all the other entries are set to empty. We also assume that node i knows the state of each outgoing link $w(i,k), k \in N^i$.

When node i receives an input event at time t, node i behaves in one of three possible ways:

(1) Node i remains idle and all distance estimates are left unchanged

(2) Node i receives LD^k_j from neighbor k, updates the estimates LD^i_{jk} and leaves all other estimates unchanged

(3) Node i updates $S^i_j(t)$ and $FLD^i_j(t)$ for destination j based on the following equation

$$S^i_j(t) = \{k | LD^i_{jk}(t) \prec FLD^i_j(t),\ k \in N^i\} \tag{4}$$

and updates its feasible logical distance as follows

$$FLD^i_j(t) = min\left(LD^i_{jk}(t) \oplus ld(i,k)(t)\right) \tag{5}$$

for all LD^k_j reported by each neighbor k and over all neighbors in N^i.

In Eqs. (4) and (5), $ld(i,k)$ is the logical distance of the adjacent link $l_{i,k}$, and $\oplus$ is the binary operator defined by f^p, which is used to combine two paths (or links) and compute the logical distance of the resulting composite path.

In addition, node i refreshes the logical distance of each non-dominated path maintained for j (up to x non-dominated paths), and sends neighbors updates if any change occurs; otherwise, it leaves all other estimates unchanged.

4.2 Loop-freedom in DMR

Loop-freedom is well understood in the context of shortest-path routing (e.g., [17–19]), and in DMR we extrapolate the inherent ordering of loop-free shortest-path routing to the case of logical distances.

A set of logical distances $\mathcal{D}$ is total-ordered w.r.t. $\preceq$ if the following four

properties are satisfied: (1) $\preceq$ is reflexive, i.e., $a \preceq a, \forall a \in \mathcal{D}$; (2) $\preceq$ is anti-symmetric, i.e., if $a \preceq b$ and $b \preceq a$, then $a = b$; (3) $\preceq$ is transitive, i.e., if $a \preceq b$ and $b \preceq c$, then $a \preceq c$; and (4) for $\forall a, b \in \mathcal{D}$, either $a \preceq b$ or $b \preceq a$. $a \prec b$ denotes that $a \preceq b$ while $a \neq b$.

For example for WSP, if $LD(p)$ and $LD(q)$ are the logical distances of two paths p and q, respectively, then $LD(p) \prec LD(q)$ only if $D^p < D^q \vee (D^p = D^q \wedge B^p > B^q)$; for LCSP, $LD(p) \prec LD(q)$ only if $D^p < D^q \vee (D^p = D^q \wedge C^p < C^q)$.

When only a single additive metric is considered, $\preceq$ and $\prec$ reduce to the normal $\leq$ and $<$; and Eq. (4) reduces to the source node condition (SNC) in [17], which considers path distances in positive real numbers only, on which the total order $\leq$ and $<$ are always well defined. The proof that using Eq. (4) to change successor sets cannot lead to loops is presented elsewhere [20] in the context of selecting an x-optimal path set, and is omitted here for brevity.

4.3 Routing optimization

The aggregate of the routing entries for destination j maintained at each node forms a directed graph rooted at j, which is a subgraph of network G and denoted by SG_j that includes links $\{l_{i,k} | k \in S_j^i \text{ for } \forall i \in V\}$. If routing converges correctly, SG_j is a directed acyclic graph (DAG) in which each node can have multiple successors for node j.

Although multiple SG_j can exist for the destination j, to achieve routing optimization in constrained path computation, DMR constructs SG_j in a way that the path with shortest logical distance for destination j is always maintained (by Eq. (4) and (5)), and as such makes SG_j an optimal successor graph w.r.t. the given optimization metric. The multiple paths computed between node i and destination j are called the *logical shortest multipath*, denoted by LSM_j^i, and is such that at least one of the paths in it has the minimal logical distance for j.

Like the distributed Bellman-Ford (DBF) algorithm [16], DMR communicates logical distances only amongst neighboring nodes, and therefore avoids expensive routing overhead caused by disseminating link-state information throughout the network. However, major differences exist between DMR and DBF. First, DMR can optimize the routing as defined by an abstract optimization function f^p, as well as the simple metrics used in best-effort routing. Second, DMR maintains x non-dominated paths for each destination, and therefore is also able to support multi-constrained path selection. Lastly, DMR is a loop-free QoS routing protocol due to the use of Eqs. (4) and (5).

4.4 Handling Network Dynamics

Even though Eqs. (4) and (5) guarantee loop-free non-dominated paths, it may be the case that node i is un-

able to find a neighbor k that has reported a logical distance that is smaller than the feasible logical distance (FLD_j^i) maintained at node i. When this occurs, node i must increase FLD_j^i in order to choose a new set of successors, but in a way that all nodes whose LSMs for j involve node i have incorporated that update in their own computations of logical distances and feasible logical distance.

DMR uses diffusing computations to accomplish the above task by coordinating node i with all upstream nodes that use node i in their LSMs for destination j. Accordingly, nodes running DMR can be in two states: *ACTIVE* or *PASSIVE*. A node i is in passive state if its successor set S_j^i given by Eq. (4) includes nodes that can provide the optimal path with the shortest logical distance for j, i.e., the multiple paths between nodes i and j form an LSM. Nodes in passive state behave much like DBF, i.e., they simply use Eq. (5) to compute the shortest logical distance for destination j, without having to establish any coordination with their neighbors.

Node i becomes active when none of its neighboring nodes in S_j^i can provide the optimal path for destination j. At this point, node i must synchronize with its upstream nodes before it is allowed to increase its logical feasible distance in order to obtain a new set of successors. To become active, node i originates a diffusing computation by sending each neighbor a *QUERY* that reports the desired logical distance for destination j. Node i detects the termination of this diffusing computation when it has received a *REPLY* from each neighboring node. After that, node i can be sure that all the upstream nodes using i for destination j either are no longer in the successor graph SG_j, or have incorporated the new logical distance that node i reported in its query, and as such is free to raise its feasible logical distance to the desired value.

Node i returns to passive state if at least one of the newly obtained successors (by Eq. (4)) provides the shortest logical distance for destination j; otherwise, node i immediately starts another diffusing computation by sending out new queries and continues to stay in active state. The mechanism used by DMR ensures that node i can return to passive state after a finite number of active phases. After becoming passive, node i sends each neighboring node an *UPDATE* to announce the changes that occur to its routing table, if there is any. For further details of diffusing computations, readers may refer to [17].

4.5 Path-Weight Propagation and Deduction

For each non-dominated path p computed for destination j, besides the logical distance, its raw path weight w^p must also be maintained, because we need w^p to verify whether p can be a feasible path when a request to forward traffic arrives. DMR exploits the same routing messages used for

the exchange of logical distances to track w^p, and as such incurs no more communication overhead.

Given that C constraints are being considered, the distance vector reporting a path p for destination j by node k is a tuple of $\{j, LD_j^k, pw_j^k\}$, in which LD_j^k and $pw_j^k = \{pw_j^k(1), ...pw_j^k(m)...pw_j^k(C)\}$ are the logical distance and the associated path weight for p, respectively. Provided that $pw_j^j(m) = 0$ for an additive metric and $pw_j^j(m) = \infty$ for a minimal metric, then after receiving the distance vector from neighbor k via adjacent link $l_{i,k}$, whose link weight is $w(i,k) = \{w_1, ...w_m, ...w_C\}$, node i can compute the corresponding path weight for j by

$$pw_j^i(m) = \begin{cases} pw_j^k(m) + w_m & \text{add. metric} \\ min\left(pw_j^k(m), w_m\right) & \text{min. metric} \end{cases} \quad (6)$$

where $m = 1, 2...C$.

Assume that the minimal logical distance reported by neighbor k for destination j at node i is $\widetilde{LD}_{jk}^i$, and that the current feasible logical distance for j at node i is FLD_j^i. According to Eq. (4), path $l_{i,k} \circ p$ (operator $\circ$ is used to concatenate two paths or links) is now considered as a candidate path for j if

$$\widetilde{LD}_{jk}^i \prec FLD_j^i \ \ (i.e., \ k \in S_j^i) \quad (7)$$

Path $l_{i,k} \circ p$ can be upgraded to a non-dominated path either if not being dominated by any path currently maintained for j, or by removing existing paths that can be dominated by $l_{i,k} \circ p$.

4.6 Example

Figure 2 demonstrates how nodes that run DMR exploit distance vectors to deduce their non-dominated path sets for the destination j without knowing global network state.

In Figure 2 (a), solid arrows mark the logical shortest multipath (LSM_j^i) from node i to j, in which each node is labeled with (LD, FLD), i.e., its shortest logical distance and feasible logical distance for j; and each link is labeled with the associated logical distance as calculated by a given f^p (for simplicity, we assume that the logical distances used in this example are positive real numbers and operator $\oplus$ is the simple addition operator $+$). According to $S_j^i = \{n | \widetilde{LD}_{jn}^i \prec FLD_j^i, n \in N^i\}$ (Eq. (4)), nodes i, a and e can have multiple feasible successors that can be used to forward traffic for j, and the neighbors which provide the shortest logical distances for destination j are included in the successor sets S_j^i, S_j^a and S_j^e, respectively.

Figure 2 (b), Figure 2 (c) and Figure 2 (d) show how non-dominated paths can be deduced based on mechanisms introduced in Sec. 4.5, at nodes that are two hops, three hops and four hops away from j. Nodes are labeled with the non-dominated paths $p(C, D)$ that they compute for j, and links are labeled with their link weights that consist of two additive metrics (C, D), where C and D are the cost and delay, respectively. Solid arrows indicate the directions to which distance vectors are propa-

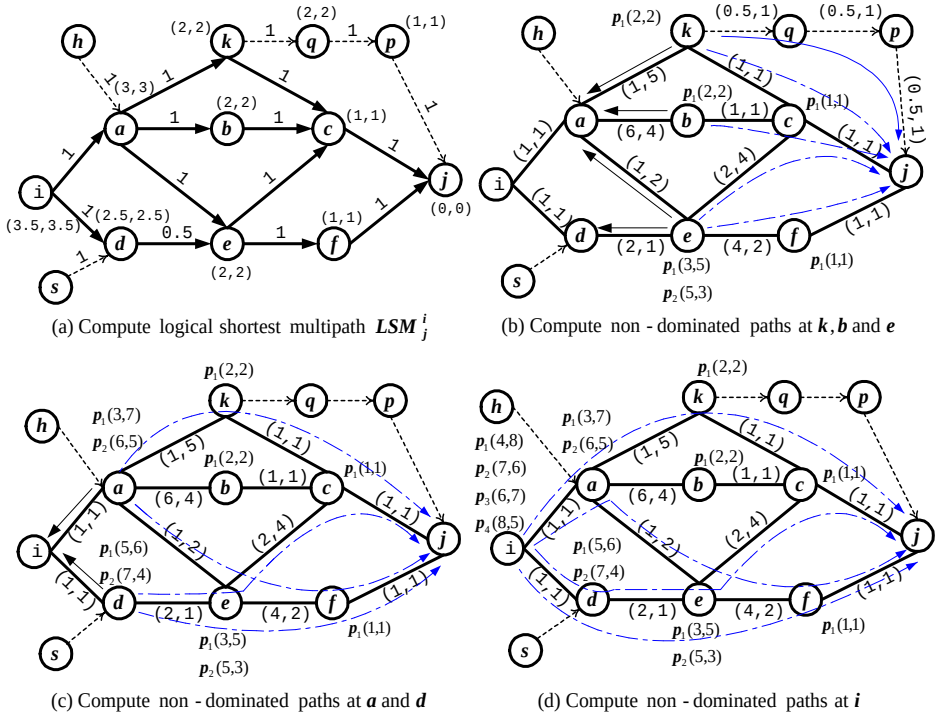

Fig. 2. Operation illustration, two additive metrics

gated upstream the logical shortest multipath LSM_j^i towards i. To be clear, each non-dominated path computed by DMR is also highlighted by a dash-dot arrow.

At node e, two paths (c, j) and (f, j) are reported from neighbor c and f, respectively, and both have the path weight $(1, 1)$. By applying Eq. (6), e obtains two paths for j: $(e, c, j) = (3, 5)$ and $(e, f, j) = (5, 3)$. Because neither dominates another, node e keeps both as the non-dominated paths for j, and reports them to its upstream neighbors a, d in the logical shortest multipath LSM_j^i, as shown in Figure 2 (b). Nodes a and d behave similarly after receiving the paths reported by their successors, and derive two non-dominated paths for j accordingly. Though node a receives four paths (one from each of k and b, and two from e), path $(a, b, c, j) = (8, 6)$ is dominated by path $(a, e, f, j) = (6, 5)$, and path $(a, e, c, j) = (4, 7)$ is dominated by path $(a, k, c, j) = (3, 7)$, as such are not inserted into the non-dominated path set of node a, as depicted by Figure 2 (c). Because Eq. (4) is not satisfied, path (k, q, p, j) cannot be used by node k for destination j, though its weight (1.5, 3) is not dominated by another path $(k, c, j) = (2, 2)$, as shown by Figure 2 (b).

Node i finally can have four non-dominated paths for destination j, as depicted in Figure 2 (d): $(i, a, k, c, j) = (4, 8)$, $(i, a, e, f, j) = (7, 6)$, $(i, d, e, c, j) = (6, 7)$ and $(i, d, e, f, j) = (8, 5)$. Note that the number of paths maintained by node i is more than the number of successors maintained in S_j^i. This cannot be achieved by using prior distance-vector based multipath routing pro-

tocols such as DASM [19] and MDVA [18], and is accomplished by DMR without knowing complete network topology.

4.7 Optimization Function f^p

In practice, the choice of f^p is policy-oriented or application-oriented, because there does not exist an absolutely better or best routing optimization metric for a given network. Therefore, the strategy adopted by DMR is to specify the properties that an optimization function must have, instead of specifying a specific f^p. The advantage of this approach is that, diversified routing optimization policies can be defined and implemented, and the convergence of DMR is also ensured simultaneously if f^p is properly selected.

More specifically, as long as the f^p being used is both **monotone and isotone** [1], within finite time after the last link-state change occurs in the network, DMR converges correctly, and maintains the optimal path for each destination. The specification and correctness proof of DMR are presented in Appendices A, B.

The reason that f^p must be monotone and isotone for DMR to converge correctly is that, unlike the metrics used in best-effort routing, QoS routing often requires that the paths for a destination be optimized w.r.t. an abstract function f^p, for which the *sub-optimal structure* property does not always hold. Sub-optimal structure means that any subpath of a shortest path between source and destination is also the shortest between the two end nodes of this subpath. Consequently, the total order relation $\preceq$ between two paths may not be preserved after extending them by a third common path. This can be illustrated by the examples depicted in Figure 3. We assume that nodes used in our discussion run a DBF-like distance-vector protocol.

Unlike WSP, shortest-widest path (SWP) routing attempts to find the path having the maximal bandwidth, and the one with shortest distance is selected if multiple such paths are found with the same bandwidth. It is easy to verify that isotone is not true in SWP. In Figure 3 (a), links are labeled with (B,D) where B is the bandwidth and D is the distance (in hops), respectively. Node 1 chooses node 2 as the successor for destination j because path $(1, 2, j)$ offers more bandwidth (5M) than path $(1, j)$ (2M); node S has no other choice but node 1 as the successor for destination j. As a result, when each node selects its successor distributively, the path formed from S to j is $(S, 1, 2, j) =< 2M, 3 >$, which is not optimal because path $(s, 1, j) =< 2M, 2 >$ and $< 2M, 2 > \prec < 2M, 3 >$.

Figure 3 (b) is an example in which the monotone property does not

[1] Monotone means that the logical distance of a path cannot decrease when the path is extended, and isotone means that the order $\preceq$ between two paths must be preserved when they are prefixed or suffixed by a common third path. Readers can refer to [8] for details.

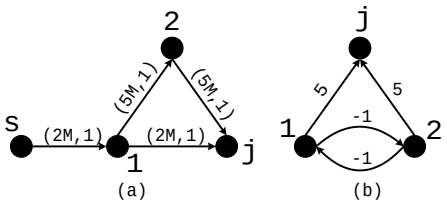

Fig. 3. Counter examples – when isotonicity or monotonicity fail

hold. Assume that link $l_{1,2}$ and $l_{2,1}$ have negative cost -1, and the f^p used is simple addition $\sum lc_i$ where lc_i is the cost of every intermediate link in the path. It is easy to see that the paths for destination j at nodes 1 and 2 can oscillate and never converge. In this simple scenario, the result is obvious, because DBF cannot converge if the topology contains negative cycles [21]. More optimization functions can be found in [4,8].

5 Complexity Analysis

At node i, provided that up to x non-dominated paths are maintained for each destination, the space complexity is $O(x|N^i|N+xN) = O(x|N^i|N)$, where $|N^i|$ is the number of neighbors of node i, because the main routing table and each neighbor table have $O(N)$ entries, and each entry can keep up to x routes for each destination. In practice, the number of non-dominated paths between source and destination can be exponential [12]. Though Yuan [12] shows that $O(N^2 \lg(N))$ non-dominated paths need to be maintained to have high probability of finding feasible paths, simulations show that DMR can achieve a satisfactory success ratio for MCP by maintaining a small number of non-dominated paths. The computation complexity is the time taken for a node to process distance vectors regarding a particular destination, and it is easy to see that it is $O(x|N^i|)$. When only the single shortest path is maintained, then the space and computation complexity reduce to $O(|N^i|N)$ and $O(|N^i|)$, respectively, which are the same as that of DBF [16].

The time complexity of DMR is the time it takes to converge after a single change in the network, and the communication complexity is the amount of messages required to propagate this change before all nodes that run DMR integrate it and update their routing tables accordingly. In practice, however, the timing and range of link changes occur in complex patterns for which closed form expressions are difficult to obtain. Section 6.1 simulates the time and communication complexity of DMR by comparing with typical distance-vector and link-state routing protocols widely used in Internet.

6 Simulation

We implemented DMR in NS2 network simulator [22]. Three types of topologies are used: ANSNET, *Pure-random* graph and *Waxman* graph. ANSNET is widely used by Chen and Nahrstedt [11] and other researchers to study QoS routing algorithms. In Pure-random graphs, the existence of the link between any two nodes is determined by a constant probability $\{p_r|\ 0 < p_r < 1\}$, while in Wax-

man graphs, p_r is defined by $p_r = \alpha e^{-\frac{d}{\beta L}}, 0 < \alpha, \beta < 1$, where d is the distance between two nodes and L is the maximal internodal distance in the graph, such that geographical location is also accounted for.

For all configurations, each link weight component is uniformly distributed in $[10, 20]$, except for ANSNET with two constraints, in which the first component is uniformly distributed within $(0, 50]$, while the second is within $(0, 200]$ [1]. Each source-destination pair is randomly chosen from the networks, and all results presented here are averaged over 5000 randomly generated routing requests, for different ranges of routing constraints. Routing metrics being considered are additive unless specified otherwise, because most algorithms proposed for MCP, and those being used to compare with DMR in our simulations, only address additive constraints. The f^p used by DMR is a linear combination that considers each link-weight component equally, which is defined by $f^p = \sum \alpha w_i, \alpha = 1/k$, if there are k constraints. It is easy to verify that this function is both monotone and isotone.

6.1 Routing Complexity

We compare DMR with the distance-vector (DV) routing protocol based on DBF [16], and the link-state (LS) routing protocol based on Dijkstra's algorithm (using ANSNET). Both DV and LS used in the simulation are the original implementations in NS2 and with all the pre-configured parameters unchanged. The types of input events studied are: (1) routing establishment (i.e., establishing routes for all destinations at each node for the first time since nodes are powered on), (2) single link failure, (3) single node failure, and (4) a sequence of link failures. We only simulate link or node failure because they are the extreme situations that may cause logical distances to increase or network partitions, which are the worst case for schemes based on distance vectors.

For each protocol, we record the *operation count*(OC), which is the iterations of the main loops in its implementation (sum over all nodes in the network); the *message count (MC)*, which is the total number of messages propagated over all links (both operation count and message count are measured since the start of the simulation until all nodes stop updating their routing tables); and the *convergence time (CT)*, which is the time since the start of an event until all nodes stop updating their routing tables. For a single link or node failure, multiple nodes and links are randomly chosen; and for a sequence of link failures, multiple links are randomly selected to go down at different simulation time instants. Table 1 presents results for routing establishment, single link and single node failure, and Table 2 presents the result when a sequence of links fail. For each evaluation metric, we show

[1] The purpose of such configuration is to compare with existing simulation results of MCP algorithms, e.g. in [23].

Table 1
Complexity comparison – single event, ANSNET

	Route establishment		
	DMR	DV	LS
CT	128±11 (ms)	192±43 (ms)	523±5 (ms)
MC	6029±466	1690±22	4945±14
OC	66454±5532	52691±1501	46331±165
	Link failure		
	DMR	DV	LS
CT	90±46 (ms)	50±8 (ms)	61±5 (ms)
MC	6070±499	1820±131	5295±5
OC	66712±5318	55840±2411	55204±37
	Node failure		
	DMR	DV	LS
CT	192±25 (ms)	329±3 (ms)	62±6 (ms)
MC	6341±323	27728±4266	5293±5
OC	69443±3649	2210758±329170	54951±84

both the mean and the standard deviation of the collected samples.

For route establishment, DMR has the best CT and similar OC compared to LS and DV. DMR sends more updating messages than LS or DV does (though they are still in the same order) because in the current implementation of DMR, each message contains only one distance vector. The communication overhead of DMR can be further reduced by allowing multiple vectors in each updating message. LS takes the longest time to stop updating routing tables because nodes that run LS need to conduct the Dijkstra's algorithm on the whole topology for every newly received link-state update, while nodes that run DMR or DV only update routing entries for the destination being reported by each message. DMR and DV behave similarly in the case of a single link failure, because the degree of ANSNET is more than one, and no network partition occurs when only one link fails. However, due to the counting-to-infinity problem, DV incurs a large number of steps when a single node fails. This is the reason why the MC and OC of DV are one or two orders of magnitude more than that of DMR and LS. Though the DV in NS2 uses a heuristic - *split horizon plus reverse poison* to work against routing loops, they cannot handle routing loops properly.

For a sequence of link failures, if links are chosen such that a node loses all its adjacent links at some point, DV again behaves poorly due to counting-to-infinity; otherwise, its performance is comparable to DMR and LS. This is the reason why the standard deviations of DV's MC and OC are high. Because the last link failure may not be the cause for a node to loose all its adjacent links, the CT of DV on average can be less than that of DMR and LS. We observe that the MC, OC and CT

Table 2
Complexity comparison – multiple events, ANSNET

	A sequence of link failures		
	DMR	DV	LS
CT	104±83 ms	92±36 ms	106±30 ms
MC	7553±764	53670±44038	8347±32
OC	81894±8043	3993506±3022200	91198±202

of LS start exceeding that of DMR. This is intuitive because more link-state updates need to be sent out to keep nodes updated about the current network state, and the routing tables also need to be refreshed more frequently in order to cope with the latest changes. The more link failures occur, the worse LS performs. This further proves that having timely global network state at each node is not a good approach in practice, especially when the network state changes often, which usually is the case in QoS routing.

The above results clearly show that DMR can provide differentiated QoS provisioning with better or comparable routing complexity than shortest-path routing protocols designed for best-effort traffic only.

6.2 Routing Success Ratio of Constrained Path Computation

We compare DMR against *centralized* algorithms that require complete network-state information, and show that DMR attains similar or better performance than them. Three evaluation metrics are presented: success ratio (SR), existence percentage (EP) and competitive ratio (CR), which are first introduced by

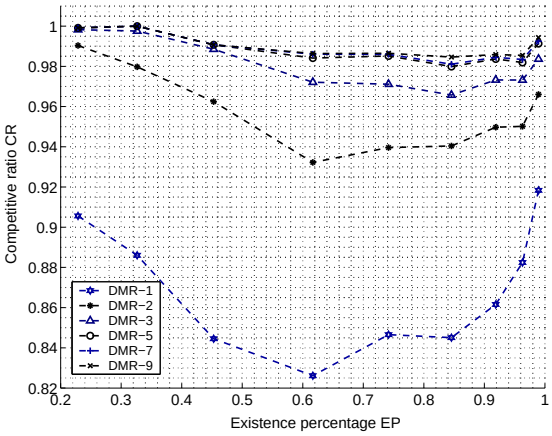

Fig. 4. ANSNET, three additive constraints

Yuan [12], and defined as follows

$$SR = \frac{routing\ requests\ being\ routed}{total\ routing\ requests} \tag{8}$$

$$EP = \frac{requests\ routed\ by\ exact\ alg.}{total\ routing\ requests} \tag{9}$$

$$CR = \frac{requests\ routed\ by\ heuristic\ alg.}{requests\ routed\ by\ exact\ alg.} \tag{10}$$

EP equals the SR of an exact algorithm, and indicates how difficult it may be to find a feasible path that meets the given request (a small EP means that it is hard to find a feasible path for the given constraints); CR indicates how well a heuristic algorithm can work in comparison to the exact algorithm with the same EP.

Table 3
SR comparison, each constraint is uniformly distributed $\sim [x_1, x_2]$, ANSNET (32 nodes 54 links)

	Exact	DMR	EDFS	JSP	KKT
$c_1 \sim [50, 65], c_2 \sim [200, 260]$	0.2476	0.2476	0.2476	0.2442	0.2476
$c_1 \sim [75, 90], c_2 \sim [300, 360]$	0.4846	0.4790	0.4836	0.4632	0.4838
$c_1 \sim [100, 115], c_2 \sim [400, 460]$	0.7050	0.6974	0.6918	0.6776	0.6992
$c_1 \sim [125, 140], c_2 \sim [500, 560]$	0.9026	0.8942	0.8892	0.8718	0.8952
$c_1 \sim [150, 165], c_2 \sim [600, 660]$	0.9740	0.9714	0.9564	0.9524	0.9740

Table 4
SR comparison, Pure-random graph (39 nodes 75 links)

	Exact	DMR	EDFS	JSP	KKT
$c_1, c_2 \sim [10, 20]$	0.1254	0.1254	0.1254	0.1230	0.1254
$c_1, c_2 \sim [20, 30]$	0.4530	0.4442	0.4502	0.4242	0.4488
$c_1, c_2 \sim [30, 40]$	0.6378	0.6282	0.6314	0.5920	0.6236
$c_1, c_2 \sim [40, 50]$	0.9120	0.9056	0.8974	0.8830	0.9088
$c_1, c_2 \sim [50, 60]$	0.9962	0.9942	0.9898	0.9908	0.9956

Table 5
SR comparison, Waxman graph (40 nodes 95 links)

	Exact	DMR	EDFS	JSP	KKT
$c_1, c_2 \sim [10, 20]$	0.1826	0.1806	0.1822	0.1752	0.1812
$c_1, c_2 \sim [20, 30]$	0.5046	0.4976	0.5042	0.4798	0.4986
$c_1, c_2 \sim [30, 40]$	0.8542	0.8470	0.8470	0.8168	0.8448
$c_1, c_2 \sim [40, 50]$	0.9956	0.9944	0.9928	0.9812	0.9906

Figure 4 illustrates how the performance of DMR varies with the size of the non-dominated path set (i.e., the parameter x) maintained for each destination, using ANSNET as the topology. We observed that (1) a satisfying SR can be achieved when each node maintains a small size of non-dominated paths, because five non-dominated paths is sufficient to achieve SRs no less than 98%, for all ranges of constraints (EP ranges from 0.23 to 0.99); and (2) the CR of DMR approaches to a saturation level and cannot be improved much by increasing the value of x (the curves of routing with five, seven and nine non-dominated paths are almost overlapped). The main reason is that, satisfying Eq. (4) can also exclude some feasible paths because of the f^p being used (see Figure 2 for an example). In what follows, the size of non-dominated path set is set to $x = 5$, unless specified otherwise.

Tables 3, 4 and 5 present SR comparisons of routing with **two** constraints amongst EDFS [7], KKT (Korkmaz, Krunz and Tragoudas [23]) and JSP (a variation of Jaffe's algorithm [10]: Dijkstra's shortest path algorithm w.r.t. the aggregated link-cost function $w(u,v) = [\frac{w_1(u,v)}{c_1} + \frac{w_2(u,v)}{c_2}]$, where c_i is the constraint on metric w_i), using ANSNET, Pure-random and Waxman graphs, respectively. As the baseline, we also implement

an exact algorithm, which has exponential running time, but can give all feasible paths, if there is any.

We observe that JSP lags behind all the other algorithms in all scenarios, which further confirms that paths optimized w.r.t. a single aggregated metric may not be a good approach for constrained path computation. EDFS outperforms DMR and other algorithms when the constraints are tight, and performs worse when the constraints are moderate and loose, for which EDFS has to iterate over more exploring sequences to achieve a better SR. Though both KKT and DMR perform consistently and similarly well, DMR solves general k-constrained MCP problems (for both additive and minimal constraints), while KKT only deals with two additive constraints and its running time is unpredictable when constraints become tight [7].

7 Conclusion

DMR is a distributed QoS routing protocol that uses distance vectors to solve MCP problems without global network state, guarantees loop freedom at any instant and last but not last, considers routing optimization as well as constrained path computation. Simulations show that DMR outperforms the shortest-path routing protocols used in today's Internet, in terms of communication overhead and running complexity; also achieves a satisfactory routing success ratio in comparison with other popular centralized MCP algorithms.

A Specification of DMR

Notation:

j: a particular destination $j \in V$

lc_k^i: the logical cost of the adjacent link $l_{i,k} \in L$; it equals to ∞ for a non-existent or failed link

N^i: the set of neighbors connected through an operational link with node i, i.e., $N^i = \{k | l_{i,k} \in L, lc_k^i \neq \infty\}$

S_j^i: the successor set chosen by node i for j, and $S_j^i \subseteq N^i$

QS_j^i: the set of neighbors for which a query has been received but a reply has not been sent

$state_j^i$: the state of node i for j, is either $ACTIVE$ or $PASSIVE$

LD_j^i: the logical distance for j as known by node i

$iPSet_j^i$ the non-dominated path set for j maintained by node i

SLD_j^i: the shortest LD for j through S_j^i

RLD_j^i: the logical distance for j that node i reports to all its neighbors

FLD_j^i: the feasible logical distance used by node i to evaluate whether *o-LFC* can be satisfied when choosing S_j^i

$\widetilde{LD}_{jk}^i$: the minimal logical distance from neighbor k to j as known by i

$iPSet_{jk}^i$: the non-dominated path set from neighbor k to j as known by i

r_{jk}^i: this flag is true if node i has sent a query for j to neighbor k but has not received a reply from k, and false otherwise

$p = p^1 \oplus p^2$ means that path p is formed by concatenating another two paths p^1 and p^2; $iPSet \leftarrow iPSet \uplus p$ means that path p is inserted into the non-dominated path set $iPSet$ if p is not dominated by any existing path in $iPSet$

For each destination j, node i maintains a **routing table** that contains LD_j^i, S_j^i, $iPSet_j^i$, FLD_j^i, RLD_j^i, QS_j^i, $state_j^i$ and SLD_j^i; a **distance table** that contains $L\widetilde{D}_{jk}^i, \forall k \in N^i$; a **link table** that includes $lc_k^i, \forall k \in N^i$ and $iPSet_{j,k}^i$; and a **flag** r_{jk}^i for each outstanding query for j.

Pseudo-code at node i:

1: **procedure** INITIALIZE_NODE(j)
2: $L\widetilde{D}_{jk}^i \leftarrow \infty$
3: $r_{jk}^i \leftarrow false$
4: $LD_j^i \leftarrow \infty$ ▷ except $LD_i^i \leftarrow \bar{0}$
5: $SLD_j^i \leftarrow \infty$ ▷ except $SLD_i^i \leftarrow \bar{0}$
6: $FLD_j^i \leftarrow \infty$ ▷ except $FLD_i^i \leftarrow \bar{0}$
7: $RLD_j^i \leftarrow \infty$ ▷ except $RLD_i^i \leftarrow \bar{0}$
8: $S_j^i \leftarrow \phi$ ▷ except $S_i^i = \{i\}$
9: $QS_j^i \leftarrow \phi$
10: $state_j^i \leftarrow PASSIVE$
11: **for** each $n \in N^i$ **do**
12: send $\{UPDATE, i, \bar{0}, <null>, i\}$ to n
13: **end for**
14: **end procedure**

1: **procedure** UPDATE_SUCCESSOR_SET(j)
2: $S_j^i \leftarrow \{n | L\widetilde{D}_{jn}^i \prec FLD_j^i | n \in N^i\}$ ▷ *o-LFC*
3: **for** each $s \in S_j^i$ **do**
4: **for** each $p \in iPSet_{j,s}^i$ **do**
5: $p \leftarrow p \oplus l_{i,s}$ ▷ path deduction
6: $iPSet_j^i \leftarrow iPSet_j^i \uplus p$ ▷ consider multiple non-dominated paths
7: **end for**
8: **end for**
9: $SLD_j^i \leftarrow min\{L\widetilde{D}_{js}^i \oplus lc_s^i | s \in S_j^i\}$ ▷ $SLD_j^i \leftarrow \bar{\infty}$ if $S_j^i = \phi$
10: $w_{SLD_j^i} \leftarrow$ *the path weight associated with* SLD_j^i
11: **for** each $n \in N^i \wedge n \bar{\in} S_j^i$ **do**
12: $iPSet_{jn}^i.clear()$ ▷ avoid stale path information
13: **end for**
14: **end procedure**

1: **procedure** PROCESS_VECTOR($mt, k, rld, <d^p, w^p>, j$)
2: **if** $j = i$ **then**
3: **if** $mt = QUERY$ **then**
4: send $\{REPLY, i, \bar{0}, <null>, j\}$ to k
5: **else**
6: return
7: **end if**
8: **end if**
9: **if** $mt = REPLY$ **then**
10: $r_{jk}^i \leftarrow false$
11: **end if**
12: $iPSet_{j,k}^i \leftarrow iPSet_{j,k}^i \uplus p$ ▷ $p = <d^p, w^p>$
13: $L\widetilde{D}_{jk}^i \leftarrow rld$
14: **update_successor_set**(j)
15: $LD_j^i \leftarrow min\{L\widetilde{D}_{jn}^i \oplus lc_n^i | n \in N^i\}$ ▷ Bellman-Ford equation
16: $w_{LD_j^i} \leftarrow$ *the path weight associated with* LD_j^i
17: **if** $state_j^i = PASSIVE$ **then** ▷ in passive
18: **if** $LD_j^i \prec SLD_j^i \vee S_j^i = \phi$ **then** ▷ S_j^i does not provide optimal path
19: $state_j^i \leftarrow ACTIVE$ ▷ become active
20: **if** $mt = QUERY$ **then**
21: $QS_j^i \leftarrow QS_j^i \cup k$
22: **end if**
23: $RLD_j^i \leftarrow SLD_j^i$
24: **for** each $n \in N^i$ **do**
25: $r_{jn}^i \leftarrow true$
26: send $\{QUERY, i, RLD_j^i, < SLD_j^i, w_{SLD_j^i} >, j\}$ to n
27: **end for**
28: **else** ▷ stay in passive
29: $state_j^i \leftarrow PASSIVE$
30: $FLD_j^i \leftarrow min\{LD_j^i, RLD_j^i\}$ ▷ FLD may decrease
31: **update_successor_set**(j)
32: $flag \leftarrow (RLD_j^i \neq LD_j^i)\ ?\ true : false$
33: $RLD_j^i \leftarrow LD_j^i$
34: **for** each $n \in N^i$ **do**
35: **if** $n = k \wedge mt = QUERY$ **then**
36: send $\{REPLY, i, RLD_j^i, < LD_j^i, w_{LD_j^i} >, j\}$ to n
37: **else if** $flag = true$ **then**
38: send $\{UPDATE, i, RLD_j^i, < LD_j^i, w_{LD_j^i} >, j\}$ to n
39: **end if**
40: **end for**
41: **end if**
42: **else if** $state_j^i = ACTIVE \wedge r_{jn}^i = false, \forall n \in N^i$ **then** ▷ receive all replies
43: $FLD_j^i \leftarrow min\{LD_j^i, RLD_j^i\}$ ▷ FLD can increase only at the end of an active phase
44: **update_successor_set**(j)
45: **if** $(LD_j^i \prec SLD_j^i) \vee (S_j^i = \phi \wedge RD_j^i \prec \bar{\infty})$ **then**
46: $state_j^i \leftarrow ACTIVE$ ▷ start a new active phase
47: **if** $mt = QUERY$ **then**

```
48:                 QS_j^i ← QS_j^i ∪ k
49:             end if
50:             RLD_j^i ← SLD_j^i
51:             for each n ∈ N^i do
52:                 r_jn^i ← true
53:                 send {QUERY, i, RLD_j^i, <
                        SLD_j^i, w_SLD_j^i >, j} to
                        n
54:             end for
55:         else
56:             state_j^i ← PASSIVE    ▷ return
                    to passive
57:             flag ← (RLD_j^i ≠ LD_j^i) ? true :
                    false
58:             RLD_j^i ← LD_j^i
59:             for each n ∈ N^i do
60:                 if n ∈ QS_j^i ∨ (n = k ∧ mt =
                        QUERY) then
61:                     send {REPLY, i, RLD_j^i, <
                            LD_j^i, w_LD_j^i >, j}
                            to n
62:                 else if flag = true then
63:                     send {UPDATE,i,RLD_j^i, <
                            LD_j^i, w_LD_j^i >, j}
                            to n
64:                 end if
65:             end for
66:             QS_j^i ← φ
67:         end if

68:     else                                ▷ in active
69:         if mt = QUERY then
70:             if k ∈ S_j^i ∧ RLD_j^i ≺ SLD_j^i then
71:                 QS_j^i ← QS_j^i ∪ k
72:             else
73:                 send {REPLY, i, RLD_j^i, <
                        LD_j^i, w_LD_j^i >, j} to k
74:             end if
75:         end if
76:     end if
77: end procedure
```

B Correctness Proof of DMR

A routing protocol is correct if the following two properties are satisfied: (1)*(Safeness)* the paths computed for each destination are loop-free at any time instant; (2) *(Liveness)* finite time after a sequence of network changes, the shortest distance for each destination maintained by each node converges to the correct value.

From the description in Sect. 4, it is clear that DMR consists of two mutually exclusive states: PASSIVE or ACTIVE. In passive state, DMR behaves much like DBF and uses the Bellman-Ford equation to compute the shortest logical distance for each known destination in the network; and nodes stay in passive as long as the logical distance to destination remains unchanged or decreases. Therefore, the proof of the correctness of DBF [16] also applies to DMR in passive state (if p^p is both monotone and isotone, see Lemma B.4). However, DMR behaves differently when the logical distance to destination increases, for which nodes send out queries and transit into active state. In active state, a node that runs DMR synchronizes with upstream nodes and raises its feasible logical distance up to a sufficient value such that another set of successors can be obtained. After that, DMR returns to passive state and again uses Bellman-Ford equation to computed shortest logical distance for each destination.

Base on the observations above, to prove the correctness of DMR requires showing that (a) every non-dominated path maintained for each destination j is loop-free at any time t (b) the time for a node to stay in active state is finite (c) finite time after a sequence of network changes, each node i that runs DMR has at least one successor which can provide the shortest logical distance for each destination j if the network is connected, and no successor otherwise.

Theorem B.1 *(Safety property) each path $p \in iPSet_j^i$ computed by DMR is loop-free at any time t.*

Proof Clearly, if $FLD_j^k \preceq RLD_j^k$, provided that $\widetilde{LD}_{jk}^i$ is properly updated at node i, we have $FLD_j^k \preceq \widetilde{LD}_{jk}^i \prec FLD_j^i$ if i chooses k as successor for j according to *o-LFC*, i.e., $S_j^i = \{k | \widetilde{LD}_{jk}^i \prec FLD_j^i\}$. As a result, inequality $FLD_j^k \prec FLD_j^i$ holds at every intermediate node in any path from i to j, which is loop-free simply because the strictly decreasing order of FLD. Therefore, when *o-LFC* is used to decide successor set S_j^i, loop-freedom can be ensured as long as the feasible logical distance of a node i is never set to a value larger than its logical distance for destination j as known by every neighbor $k \in N^i$, i.e., $FLD_j^i \preceq \widetilde{LD}_{ji}^k$, in every PASSIVE and ACTIVE phase.

Let us assume that at time t_n, node i transits from PASSIVE to ACTIVE for destination j, then the proof is by induction on t_n. Let $\widetilde{LD}_{ji}^k(t)$ denotes the minimal logical distance being reported to a neighbor $k \in N^i$ by node i at time t. Initially, all logical distance value are set to $\bar{\infty}$, hence $FLD_j^i(0) \preceq \widetilde{LD}_{ji}^k(0)$. Now assume that *o-LFC* is true until time t_n, then we have

$$FLD_j^i(t) \preceq \widetilde{LD}_{ji}^k, t \in [0, t_n) \quad \text{(B.1)}$$

At any time t, by the line 9 of *update_successor*() and the line 15 of *process_vector*(), we have $LD_j^i(t) \preceq SLD_j^i(t)$; by the line 23, 33, 50 and 58 of *process_vector*(), we have $SLD_j^i(t) \preceq RLD_j^i(t)$; by the line 30, 43 of *process_vector*(), we also have $FD_j^i(t) \preceq LD_j^i(t)$. Therefore, it is always true that

$$FLD_j^i(t) \preceq RLD_j^i(t), \ for \ \forall \ t \quad \text{(B.2)}$$

Hence at time t_n, it follows that

$$FLD_j^i(t_n) \preceq RLD_j^i(t_n) \quad \text{(B.3)}$$

Now node i becomes ACTIVE at t_n by sending queries out and a neighbor $k \in N^i$ receives a query at time $t_1 > t_n$, if $\widetilde{LD}_{ji}^k$ is always updated correctly at node k (this is the assumption on operational links, see Sect. 3), from (B.2), we have

$$FLD_j^i(t) \preceq \widetilde{LD}_{ji}^k(t), t \in [t_n, t_1) \quad \text{(B.4)}$$

Let t_2 be the time at which all replies are received at node i and the current active phase ends. Because FLD_j^i remains unchanged, and also there is no different RLD_j^i reported by i during the active phase either, from (B.3), the following statement is true

$$FLD_j^i(t) \preceq \widetilde{LD}_{ji}^k(t), t \in [t_1, t_2) \quad \text{(B.5)}$$

at time t_2, according to (B.2), we also have

$$FLD_j^i(t_2) \preceq RLD_j^i(t_2) \quad \text{(B.6)}$$

If the current active phase ends at t_2, then $FLD_j^i(t) \preceq \widetilde{LD}_{ji}^k(t), t \in [t_n, t_2]$. Otherwise, if a new active phase follows immediately, assume again that at $t_3 > t_2$ queries are sent out and at $t_4 > t_3$ all replies are received at node i, by following a similar arguing as above, we also have $FLD_j^i(t) \preceq \widetilde{LD}_{ji}^k(t), t \in [tn, t_4]$. Therefore, regardless how many back-to-back active phases in which

node i stays, DMR ensures that $FLD_j^i(t) \preceq \widetilde{LD}_{ji}^k(t), t \in [t_n, t_{n+1}]$. By induction, $FLD_j^i(t) \preceq \widetilde{LD}_{ji}^k(t)$ holds at any time t, and therefore any path computed by DMR is loop-free at all times.

Lemma B.2 *Every active phase of a diffusing computation has a finite duration.*

Proof In DMR, node i transits into ACTIVE by sending queries to its neighbors and remains in ACTIVE until all neighbors send back replies. An endless active phase happens only when some of the neighbors do not reply the outstanding query. However, this cannot be true due to the following reasons. First, a node replies to a query immediately and remains in the passive state as long as the shortest logical distance can be provided by its current successor set. Second, only the upstream nodes of node i in the successor graph SG_j can be affected by the transition of node i, i.e., they also transit into ACTIVE if shortest path for j cannot be found by local computation. By generating a sequence of nodes, from node i and all the way upstream the successor graph SG_j, because SG_j is loop-free at any time instant (as proven by Theorem B.1), we finally will reach the nodes that have not replied yet and not in the successor set for j of any node either. Because a node that is not a successor of any other node must reply to queries immediately, and the diameter of a practical network is finite, it follows by contradiction that any active phase of a diffusing computation will end in finite time.

Lemma B.3 *A node that runs DMR can have only a finite number of back-to-back active phases.*

Proof An active phase can be caused by a sequence of network changes or queries sent from neighbors. An infinite number of active phases occurs when a node i has back-to-back active phase without replying pending queries from its neighbor nodes, or experiences infinite number of changes. However, this cannot be true because of the following reasons. First, a query will be blocked by adding its sender k into QS_j^i, and therefore node i will not receive the same query from k endlessly. Second, in practice we can only have finite number of network changes in a sequence, and each node can only have finite number of neighbors from which it can receive queries or updates, a node eventually will return to passive after a finite number of back-to-back active phases. After that, all neighbor nodes having pending queries will receive their replies.

Lemma B.4 *In a stable topology G, if the f^p being used is both* monotone and isotone*, DMR always converges correctly, i.e., each node that runs DMR obtains the optimal path with shortest logical distance for each destination $j \in G$ in finite time.*

Proof According to [8], for stable topologies, a distance vector routing protocol always converges correctly w.r.t. the optimization metric being considered, if and only if the associated optimization function f^p is both monotone and isotone.

Based on the description in Sect. 4, DMR behaves similar to DBF when nodes stay in the passive state and use Bellman-Ford equation to compute the shortest logical distance for each destination (except that multiple successors are allowed if *o-LFC* is satisfied). Therefore, the convergence property established for all distance-vector based routing protocols (See [8], Sect. 6.2, Proposition 3-5) also applies to DMR. [2]

Theorem B.5 *(Liveness property) within finite time after the last state change occurs in the network, DMR converges correctly.*

Proof This follows immediately from Lemmas B.2, B.3 and B.4.

References

[1] Z. Wang and J. Crowcroft, Quality-of-SService Routing for Supporting Multimedia Applications, IEEE Journal of Selected Areas in Communications, Vol. 14, No. 7, pp 1228-1234, 1996.

[2] P. V. Mieghem, H. D. Neve and F. Kuipers, Hop-by-Hop Quality of Service Routing, Comput. Networks, Vol. 37, No. 3-4, pp 407–423, 2001.

[3] J. Wang and K. Nahrstedt, Hop-by-Hop Routing Algorithms for Premium-class Traffic In DiffServ Networks, in proceedings of IEEE INFOCOM, 2002.

[4] S. Nelakuditi, Z. Zhang, R. P. Tsang and D. H. C. Du, Adaptive Proportional Routing: A Localized QoS Routing Approach, IEEE/ACM Trans. Netw., Vol. 10, No. 6, pp 790–804, 2002.

[5] J. L. Sobrinho, Algebra and Algorithms for QoS Path Computation and Hop-by-Hop Routing in the Internet, IEEE/ACM Trans. Netw., Vol. 10, No. 4, pp 541–550, 2002.

[2] Lemma B.4 can also be proven by applying a proof similar to that of DBF [16], in which a stable topology is assumed.

[6] B. Smith and J. J. Garcia-Luna-Aceves, Efficient Policy-Based Routing Without Virtual Circuits, in proceedings of QSHINE'04, Dallas, Texas, Oct. 2004.

[7] Z. Li and J. J. Garcia-Luna-Aceves, Solving The Multi-Constrained Path Selection Problem by Using Depth First Search, in proceedings of QSHINE'05, Orlando, Florida, Aug. 2005.

[8] J. L. Sobrinho, Network Routing with Path Vector Protocols: Theory and Application, in proceedings of ACM SIGCOMM, 2003.

[9] P. V. Mieghem and F. A. Kuipers, Concepts of Exact QoS Routing Algorithms, IEEE/ACM Trans. Netw., Vol. 12, No. 5, pp 851–864, 2004.

[10] J. M. Jaffe, Algorithms for Finding Paths with Multiple Constraints, IEEE Networks, Vol. 14, pp 95-116, 1984.

[11] S. Chen and K. Nahrstedt, On Finding Multi-constrained Paths, in proceedings of ICC'98, pp 874–879, Jun. 1998.

[12] X. Yuan, Heuristic Algorithms for Multiconstrained Quality-of-Service Routing, IEEE/ACM Trans. Netw., Vol. 10, No. 2, pp 244–256, 2002.

[13] H. D. Neve and P. V. Mieghem, TAMCRA: A Tunable Accuracy Multiple Constraints Routing Algorithm, Computer Communications, Vol. 23, pp 667–679, 2000.

[14] S. Chen and K. Nahrstedt, Distributed Quality-of-Service Routing in Ad-Hoc Networks, IEEE Journal on Selected Areas in Communications, Vol. 17, No. 8, Aug. 1999.

[15] J. Doyle, Routing TCP/IP, Cisco Press, 1998.

[16] D. Bertsekas and R. Gallager, Data Networks - 2nd Edition, Prentice-Hall, 1992.

[17] J.J. Garcia-Lunes-Aceves, Loop-free Routing Using Diffusing Computations,IEEE/ACM Trans. Netw., Vol. 1, No. 1, pp 130–141, 1993.

[18] S. Vutukury and J. J. Garcia-Luna-Aceves, MDVA: A Distance-Vector Multipath Routing Protocol, in proceedings of IEEE INFOCOM, Anchorage, Alaska, USA, 2001.

[19] W.T. Zaumen and J. J. Garcia-Luna-Aceves", Loop-Free Multipath Routing Using Generalized Diffusing Computations", in proceedings of IEEE INFOCOM, San Francisco, California, USA, 1998.

[20] Z. Li and J. J. Garcia-Luna-Aceves, A Distributed Approach For Multi-Constrained Path Selection And Routing Optimization, in proceedings of QSHINE'06, Waterloo, Ontario, Canada, Aug. 2006.

[21] T. H. Cormen et al, Introduction to Algorithms, Second Edition, McGraw-Hill, 2003.

[22] NS2, the Network Simulator, http://www.isi.edu/nsnam/ns/.

[23] T. Korkmaz, M. Krunz and S. Tragoudas, An Efficient Algorithm for Finding a Path Subject to Two Additive Constraints, in proceedings of the ACM SIGMETRICS, pp 318–327, 2000.

[24] W. Liu and W. Lou and Y. Fang, An Efficient Quality of Service Routing Algorithm for Delay Sensitive Applications, Computer Networks, Vol. 1, No. 47, pp 87–104, 2004.