Using Dijkstra to Compute Hop-by-Hop QoS Paths

Bradley R. Smith
Computer Engineering Department
University of California, Santa Cruz
Santa Cruz, CA 95064

***Abstract*—The Internet is based on a hop-by-hop routing model where forwarding decisions are based on a packet's destination. With the evolution of the Internet to support real-time applications such as video and audio distribution, the Internet's routing architecture must be enhanced to support metrics that ensure the quality-of-service requirements of these applications.**

Sobrinho has shown that some of these metrics, when used with a Dijkstra routing computation, may result in paths that are not least-cost or loop-free, and don't support hop-by-hop forwarding in the general case. To address this problem Sobrinho has presented a technique for computing paths using Dijkstra that, while not directly supporting hop-by-hop forwarding, do ensure that packets forwarded based on these paths will follow loop-free, least-cost paths. However, this results in routers forwarding packets along different paths from those they computed.

In this paper we build on Sobrinho's work by presenting a sufficient condition for computing hop-by-hop paths. This condition, while not as general as Sobrinho's solution, has more relaxed requirements of the path algebra. Given this greater latitude we are then able to present a new technique for computing paths that are loop-free, least-cost, and support hop-by-hop forwarding when used with a Dijkstra route computation, thus restoring the visibility of routers into the paths they forward packets over.

I. Introduction

The Internet is based on a shortest-path, hop-by-hop routing architecture. Forwarding decisions are made at each hop based on a packet's destination and forwarding state computed independently by each node. The forwarding state is computed using a shortest-path routing algorithm that minimizes an additive weight function, typically based on a delay-related metric. It is a well-known property of shortest-paths that, for any shortest path with an embedded node x, the subpath from the source to x is also a shortest path [1]. With additive weights, this property holds when shortest-path trees are computed independently at each node.

As the Internet evolves to be the foundation of a fully converged communication infrastructure, there has been an increasing need to support real-time applications such as audio or video distribution in either streaming or conferencing formats. Real-time traffic has quality-of-service (QoS) requirements beyond the minimization of delay-related metrics, including constraints on delay jitter, loss probability, bandwidth, and combinations of these [2]. Examples of combinations include *widest-shortest* where, of the set of minimum delay paths, the path with maximum available bandwidth is chosen, and its converse, *shortest-widest*.

These metrics include non-additive components. Path bandwidth is computed as the minimum bandwidth on the links composing the path, and is a *concave* metric. Path loss probability is computed as the product of link loss probabilities, and is a *multiplicative* metric. The question then arises whether the Internet's shortest path, hop-by-hop routing architecture continues to deliver packets over loop-free, least-cost paths in the context of such QoS metrics.

In [3] Sobrinho investigated this question in the context of a Dijkstra shortest-path computation. In this work he showed that *isotonicity* is required for a Dijkstra computation to produce lightest paths in general. He further showed that without a strict version of isotonicity not every implementation of

Dijkstra would result in paths that supported loop-free hop-by-hop routing.

Lastly, he presented a modified version of the Dijkstra algorithm that would compute loop-free, least-cost paths with loosely-isotonic metrics. However, the paths computed by this enhanced algorithm were not hop-by-hop in the sense that the set of paths computed by nodes in a network that traversed a given node en route to the same destination may have conflicting forwarding decisions at that node. Said another way, the forwarding decisions required by the paths computed by Sobrinho's modified Dijkstra algorithm depended on both the source and destination of a packet, and required the output of routing computations run from the perspective of all nodes in the network. Fortunately, Sobrinho was able to show that, while the paths computed by nodes in the network may conflict in this manner, in fact the packets would be forwarded over a loop-free, least-cost path when hop-by-hop forwarding was used.

In this paper we build on Sobrinho's work by presenting a sufficient condition for computing hop-by-hop paths. This condition, while not as general as Sobrinho's solution, has more relaxed requirements of the path algebra. We then use this result to show that a new form of word-weight, that we call *lexicographically-lightest trailing subpath*, results in paths that are loop-free, least-cost, and support hop-by-hop forwarding when used with a Dijkstra route computation.

II. Foundations

In the following a network is modeled as a weighted, undirected graph $G = (V, E)$, where V and E are the vertex and edge sets, respectively. Elements of E are unordered pairs of distinct vertices in V. $A(i)$ is the set of edges adjacent to vertex i in the graph. Each link $(i, j) \in E$ is assigned a weight, denoted by $w_{i,j}$. A path is a sequence of vertices $\langle v_1, v_2, \ldots, v_n \rangle$ such that $(v_i, v_{i+1}) \in E$ for $i = 1, 2, \ldots, n-1$, and all nodes in the path are distinct. Given a path p, the subpath of p from the ith to the jth vertex of p is denoted by $p_{i,j}$. For example, if $p = \langle v_1, v_2, \ldots, v_n \rangle$ then $p_{i,j} = \langle v_i, v_{i+1}, \ldots, v_j \rangle$, and $p = p_{1,n}$. The path concatenation operator $\circ$ is defined such that, if p and q are two paths where the last vertex in p is the first vertex in q, then $p \circ q$ is the path formed by the concatenation of p with q. Lastly, the weight of a path is given by:

$$w_p = \sum_{i=1}^{n-1} w_{x_i, x_{i+1}}.$$

In the following we adopt the algebra from [3]. These definitions are repeated here for convenience. The algebra is defined as a set of weights, S, a binary operation, $\oplus$, and an order relation $\preceq$. There are two distinguished weights: $\overline{0}$ and $\overline{\infty}$. The algebra has the following properties:

P1 $(S, \oplus, \overline{0})$ is a *monoid*
- S is *closed* under $\oplus$: $a \oplus b \in S$ for all $a, b \in S$;
- $\oplus$ is *associative*: $a \oplus (b \oplus c) = (a \oplus b) \oplus c$ for all $a, b, c \in S$;
- $\overline{0}$ is an *identity*: $a \oplus \overline{0} = \overline{0} \oplus a = a$ for all $a \in S$.

P2 $\overline{\infty}$ is an *absorptive element*: $a \oplus \overline{\infty} = \overline{\infty} \oplus a = \overline{\infty}$ for all $a \in S$.

P3 $\preceq$ is a *total order* on S
- $\preceq$ is *reflexive*: $a \preceq a$ for all $a \in S$;
- $\preceq$ is *anti-symmetric*: if $a \preceq b$ and $b \preceq a$ then $a = b$;
- $\preceq$ is *transitive*: if $a \preceq b$ and $b \preceq c$ then $a \preceq c$;
- For every $a, b \in S$ either $a \preceq b$ or $b \preceq a$.

P4 $\overline{0}$ is a *least element*: $\overline{0} \preceq a$ for all $a \in S$.

P5 $\oplus$ is *loosely isotone* for $\preceq$: $a \preceq b$ implies both $a \oplus c \preceq b \oplus c$ and $c \oplus a \preceq c \oplus b$ for all $a, b, c \in S$.

And following is the definition for the strict form of isotonicity.

P5-S $\oplus$ is *strictly isotone* for $\preceq$: $a \prec b$ implies both $a \oplus c \prec b \oplus c$ and $c \oplus a \prec c \oplus b$ for all $a, b \in S$ and $c \in S - \overline{\infty}$.

For future reference we note that properties **P5-S** and **P5**, via contraposition, can be restated as follows:

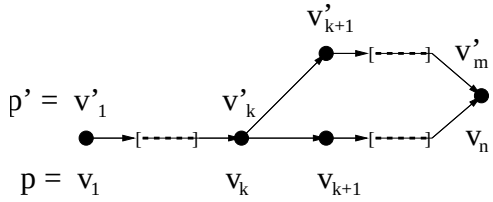

Fig. 1. Hop-by-hop and lightest subpaths.

P5-S
$$\begin{aligned}(b \prec a) \rightarrow (c \oplus b \prec c \oplus a) \quad &\leftrightarrow \\ \neg(c \oplus b \prec c \oplus a) \rightarrow \neg(b \prec a) \quad &\leftrightarrow \\ (c \oplus b \succeq c \oplus a) \rightarrow (b \succeq a) \quad &\leftrightarrow \\ (c \oplus a \preceq c \oplus b) \rightarrow (a \preceq b).\end{aligned}$$

P5
$$\begin{aligned}(b \preceq a) \rightarrow (c \oplus b \preceq c \oplus a) \quad &\leftrightarrow \\ \neg(c \oplus b \preceq c \oplus a) \rightarrow \neg(b \preceq a) \quad &\leftrightarrow \\ (c \oplus b \succ c \oplus a) \rightarrow (b \succ a) \quad &\leftrightarrow \\ (c \oplus a \prec c \oplus b) \rightarrow (a \prec b).\end{aligned}$$

The *lightest* weight path from source s to destination d is defined as a path with the least weight of all paths in the graph from s to d.

Next, we define a *hop-by-hop* path as a path where each node along the path has a 1-to-1 mapping of destination to next hop for each destination reachable from the node such that the next hop is on a lightest path to the destination. Given this foundation, the following theorem relates the hop-by-hop nature of a path to the optimality of its subpaths.

Theorem 1: A path is a hop-by-hop path *if* every subpath of the path is lightest.

Proof: By contradiction. Given two paths between the same two nodes in a graph, $p = \langle v_1, v_2, \ldots, v_n \rangle$ and $p' = \langle v'_1, v'_2, \ldots, v'_m \rangle$ where $v_1 = v'_1$ and $v_n = v'_m$, that share a leading subpath to v_k, $1 < k < Min(n, m)$ (i.e. $p_{1,k} = p'_{1,k}$ and $v_{k+1} \neq v'_{k+1}$; see Figure 1). Assume every subpath of p is lightest, but that p is not hop-by-hop. Let vertex v_k be the vertex whose next hop on p is not on the lightest path to v_n, and let the path $p'_{k,m}$ be a lightest path from v_k to v_n. This implies that $w_{p'_{k,m}} \prec w_{p_{i+k-1,n}}$, which contradicts the assumption that every subpath of p is lightest. ■

This theorem provides a concrete criteria for a routing algorithm and path algebra to ensure they compute paths that support hop-by-hop forwarding. Note that this theorem does not depend on the isotonicity of the algebra. The task for the designer of a new routing function is to prove that all subpaths of paths computed by the function are optimal, and that these paths are lightest and loop-free. The next section presents a new algebra, and shows that it satisfies these properties when used in a Dijkstra path computation.

III. Dijkstra and Hop-by-Hop

In [3] Sobrinho showed that a Dijkstra computation using loosely isotonic metrics does not compute hop-by-hop paths, but that such metrics can be used as the basis of a computation that computes routes such that, when packets are forwarded hop-by-hop, they travel along loop-free, least-cost paths. This is a subtle distinction, and is based on the fact that, with lexicographically-lightest metrics, packets may be forwarded along paths that differ from the path computed at the source.

In this section we present a new form of word-weight that, when used with a Dijkstra path computation, results in paths that are least-cost, loop-free, and support hop-by-hop forwarding in the stronger sense that packets actually traverse the path computed by the source. First, we reproduce Sobrinho's definitions for the terms *L-lightest* and *word-weight*. We then present a new form of lightest path, that we call *L-lightest-TS*, and show that, when used with a Dijkstra path computation, it results in loop-free, least-cost paths.

Consider the set of weights S as an alphabet, and let S^* be the set of all words over S. Given two words $\alpha = \alpha_1\alpha_2 \ldots \alpha_n$ and $\beta = \beta_1\beta_2 \ldots \beta_m$, Sobrinho defined the relation *lexicographically less than*, written $\prec_L$, as $\alpha \prec_L \beta$ if either:

1) $n < m$ and $\alpha_i = \beta_i$ for $1 \leq i < n$, or
2) there is an index k, $1 \leq k \leq min(n, m)$, such that $\alpha_k \prec \beta_k$ and $\alpha_i = \beta_i$ for $1 \leq i < k$.

He then defined the *word-weight* of a non-trivial path $p = \langle v_n, \ldots, v_2, v_1 \rangle$, denoted by $\omega(p)$, as the word of S^*, the ith letter of which, $\omega_i(p)$, $1 \leq i < n$, is the weight of the subpath of p that extends from the source to its ith last node.

That is, $\omega_i(p) = w(p_{n,i})$, and thus $\omega_i(p) = w(p_{n,1})w(p_{n,2})\ldots w(p_{n,n-1})$. The word-weight of the trivial path is the empty word. Path p is a *lexicographic-lightest* (*L-lightest* for short) path from s to v if its word-weight is lexicographically less than or equal to the word-weight of any other path from s to v. That is, if $\omega(p) \preceq_L \omega(q)$ for every path q from s to v. Lastly, he showed, in Theorems 4 and 5 from [3], that a Dijkstra path computation based on the L-lightest relation computes paths that, while not themselves hop-by-hop, do result in packets being forwarded along loop-free, least-cost paths.

We now define a new form of word-weight and show that, when used with a Dijkstra shortest-path computation, it results in loop-free, least-cost, hop-by-hop paths. A *trailing subpath word-weight*, denoted $\omega^{ts}(p)$, is defined as the word of S^*, the ith letter of which, $\omega_i^{ts}(p)$, $1 < i \leq n$, is the weight of the subpath of p that extends from the ith last node to the last node in the path. That is, $\omega_i^{ts}(p) = w(p_{i,1})$ and thus $\omega^{ts}(p) = w(p_{n,1})w(p_{n-1,1}), \ldots, w(p_{2,1})$. Given this definition, path p is an *L-lightest-TS* path from s to v if $\omega^{ts}(p) \preceq_{L^{ts}} \omega^{ts}(q)$ for every path q from s to v. Note that, for L-lightest-TS word-weights, the word weight for the concatenation of two paths, e.g. $\omega^{ts}(p_{n,k} \circ p_{k,1})$, can be computed by adding $\omega_k^{ts}(p_{k,1})$ to each component of $\omega^{ts}(p_{n,k})$ and appending $\omega^{ts}(p_{k,1})$ to the result.

Theorem 2: If the generalized Dijkstra algorithm from [3] is run using the L-lightest-TS algebra then, on termination, the resulting paths will be hop-by-hop.

Proof: From *Theorem 3* of [3] we know that Dijkstra computes shortest paths from the source to all reachable destinations in a graph. In the following we show that the use of L-lightest-TS metrics with Dijkstra computes paths all of whose subpaths are optimal, and therefore, by Theorem 1, these paths support hop-by-hop forwarding. We show this in three cases. ■

Leading subpaths (subpaths starting at v_n). The example illustrated in Figure 2, based on the widest-path algebra from [3], is used to illustrate this case. Given paths $p = \langle 1, 3, 5, 6\rangle$ and $p' = \langle 1, 2, 4, 5, 6\rangle$, p's leading subpath $p_{1,5}$ is not optimal as $\omega^{ts}(p'_{1,5}) = (5, 10, 10) \prec_{L^{ts}} (5, 5) = \omega^{ts}(p_{1,5})$. However, when $p_{1,5}$ is replaced by $p'_{1,5}$ in p, the resulting path is actually worse than the original:

$$\omega^{ts}(p) = (5, 5, 5) \prec_{L^{ts}} (5, 5, 5, 5) = \omega^{ts}(p'_{1,5} \circ p_{5,6}) = \omega^{ts}(p')$$

Therefore, the use of L-lightest-TS metrics on their own do not ensure all subpaths of paths computed using such metrics are optimal. However, when used in a Dijkstra computation, the dynamics of the algorithm ensure that they are. Note that this example shows that L-lightest-TS metrics are not isotonic!

Referring to Figure 2 again, at the point in a Dijkstra computation run on this graph where routes have been selected for nodes 3 and 4, two temporary-labeled routes for paths $p'_{1,5}$ and $p_{1,5}$ would have been generated from relaxing the routes to 4 and 3, respectively. In the next iteration of of Dijkstra, path $p'_{1,5}$ would be selected for destination 5, and path $p_{1,5}$ would have been deleted from the set of temporarily labeled routes in subsequent iterations. As a result the test of $p \prec p'$ would have never occurred. This is a concrete illustration of the general result that Dijkstra actually computes a *tree* of S-lightest paths [1].

More generally, by the nature of the Dijkstra algorithm, an extension of a shortest path between two vertices (e.g. p' in Figure 2) will never be compared with the extension of a longer path between the same two endpoints (e.g. p in Figure 2). This winner-take-all nature of the Dijkstra algorithm ensure that leading subpaths are always optimal.

Trailing subpaths (subpaths ending at v_1). Assume $p = \langle v_n, v_{n-1}, \ldots, v_1\rangle$ is the shortest path from v_n to v_1 (see Figure 3), but that there is a trailing subpath $p^t = p_{k,1}$, $n > k > 1$, that is not the optimal path from v_k to v_1. Let $p^a = \langle v_m^a, v_{m-1}^a, \ldots, v_1^a\rangle$ be an alternate path between v_k and v_1 (i.e. $v_m^a = v_k$ and $v_1^a = v_1$) that is better than p^t. Let $p' = \langle v'_l, v'_{l-1}, \ldots, v'_1\rangle$ be the new path from v_n to v_1 (i.e. $v'_l = v_n$ and $v'_1 = v_1$) formed by replacing p^t with p^a in p (i.e. $p'_{l,m} = p_{n,k}$ and $p'_{m,1} = p^a$). Since $\omega^{ts}(p^a) \prec_{L^{ts}} \omega^{ts}(p^t)$ it must be that $\omega_m^{ts}(p^a) \preceq \omega_k^{ts}(p^t)$. Lastly, recalling that

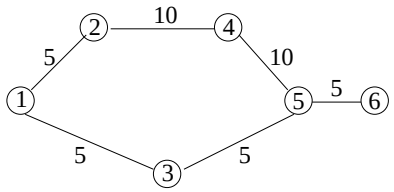

Fig. 2. Leading subpaths.

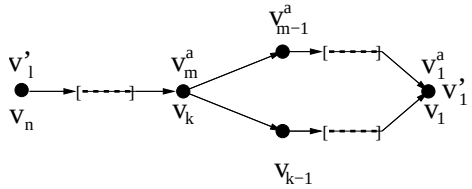

Fig. 3. Trailing subpaths.

$p'_{l,m} = p_{n,k}$, by **P5** we have, for $l \geq i > m$, that the leading $l-m$ components of $\omega^{ts}(p'_{l,m})$ and $\omega^{ts}(p_{n,k})$ are related as follows:

$$\omega_i^{ts}(p') = \omega_i^{ts}(p'_{i,m}) \oplus \omega_m^{ts}(p^a) \preceq \omega_{n-l+i}^{ts}(p_{n-l+i,k}) \oplus \omega_k^{ts}(p^t) = \omega_{n-l+i}^{ts}(p).$$

Therefore, since $\omega^{ts}(p^a) \prec_{L^{ts}} \omega^{ts}(p^t)$, it must be that:

$$\omega^{ts}(p') = \omega^{ts}(p'_{l,m} \circ p^a) \prec_{L^{ts}} \omega^{ts}(p_{n,k} \circ p^t) = \omega^{ts}(p),$$

contradicting the assumption that p is optimal.

Embedded subpaths (subpaths starting at a vertex other than v_n and ending at a vertex other than v_1). This case combines conditions from the previous two cases. First we consider the shared leading subpath prefix and the sub-optimal embedded subpath, using the reasoning from the trailing subpath case above to show that substituting a better embedded subpath results in an improved leading subpath. We then use the winner-takes-all property discussed in the first case to imply that a run of Dijkstra would only consider the newly constructed path, contradicting the assumption and completing the proof of the theorem.

Assume $p = \langle v_n, v_{n-1}, \ldots, v_1 \rangle$ is the shortest path from v_n to v_1 considered by a run of the Dijkstra algorithm (see Figure 4), but that there is an embedded subpath $p^e = p_{s,s-k}$, $n > s > s-k > 1$, that is not an optimal path from v_s to v_{s-k}. Let $p^a = \langle v_m^a, v_{m-1}^a, \ldots, v_1^a \rangle$ be an alternate path between v_s and v_{s-k} (i.e. $v_m^a = v_s$ and $v_1^a = v_{s-k}$) that is better than p^e. Let $p' = \langle v'_l, v'_{l-1}, \ldots, v'_1 \rangle$ be the new path from v_n to v_{s-k} (i.e. $v'_l = v_n$ and $v'_1 = v_{s-k}$) formed by replacing p^e with p^a in p (i.e. $p'_{l,m} = p_{n,s}$ and $p'_{m,1} = p^a$). Since $\omega^{ts}(p^a) \prec_{L^{ts}} \omega^{ts}(p^e)$ it must be that $\omega_m^{ts}(p^a) \preceq \omega_k^{ts}(p^e)$. Recalling that $p'_{l,m} = p_{n,s}$, by **P5** we have, for $l \geq i > m$, that the leading $l-m$ components of $\omega^{ts}(p'_{l,m})$ and $\omega^{ts}(p_{n,s})$ are related as follows:

$$\omega_i^{ts}(p') = \omega_i^{ts}(p'_{i,m}) \oplus \omega_m^{ts}(p^a) \preceq \omega_{n-l+i}^{ts}(p_{n-l+i,s}) \oplus \omega_k^{ts}(p^e) = \omega_{n-l+i}^{ts}(p).$$

Therefore, since $\omega^{ts}(p^a) \prec_{L^{ts}} \omega^{ts}(p^e)$, it must be that:

$$\omega^{ts}(p') = \omega^{ts}(p'_{l,m} \circ p^a) \prec_{L^{ts}} \omega^{ts}(p_{n,s} \circ p^e) = \omega^{ts}(p_{n,s-k}).$$

Based on the winner-take-all property of Dijkstra presented in the first case above, the optimality of p' implies that a run of Dijkstra would only consider extensions of p', specifically $p' \circ p_{s-k,1}$, contradicting the assumption p was the path computed by Dijkstra. ■

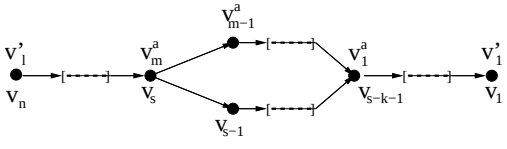

Fig. 4. Embedded subpaths.

Corollary 1: Dijkstra with the L-lightest-TS path algebra computes lightest paths.

This is a special case of Theorem 2 for the subpath composed of the full path.

Theorem 3: Dijkstra with the L-lightest-TS path algebra computes loop-free paths.

Proof: By contradiction. Assume a path computed by Dijkstra with the L-lightest-TS algebra contains a loop. Let

$p = \langle v_n, v_{n-1}, \ldots, v_k, \ldots, v_1, v_k \rangle$ be the leading subpath of this path at the point that the loop is formed at v_k, and let $p' = \langle v_n, v_{n-1}, \ldots, v_{k+1}, v_k \rangle$ be the loopfree subpath of p from v_n to v_k.

The word-weights for p' are of the form: $\omega_i^{ts}(p') = w_{p'_{i,k}}$, $n - k + 1 \geq i > 1$. The word-weights for the corresponding elements of p, expressed in terms of p', are: $\omega_{i+k-1}^{ts}(p) = w_{p'_{i,k}} + w_{p_{k,1}} + w_{1,k} = w_{p'_{i,k}} + c$.

For p to be lighter than p' (i.e. $\omega^{ts}(p) \prec_{L^{ts}} \omega^{ts}(p')$) it must either be that the length of $\omega^{ts}(p)$ (i.e. n) must be less than the length of $\omega^{ts}(p')$ (i.e. $n - k$), which is clearly not true, or that there is an i, $n - k + 1 \geq i > 1$, such that all leading components are equal, i.e. $\omega_{j+k-1}^{ts}(p) = \omega_j^{ts}(p')$, $n-k+1 \geq j > i$, and $\omega_{i+k-1}^{ts}(p) \prec \omega_i^{ts}(p)$ for the first unequal element.

By the contrapositive form of property **P5**, the last condition that $\omega_{i+k-1}^{ts}(p) = w_{p'_{i,k}} + c \prec w_{p'_{i,k}} + \overline{0} = \omega_i^{ts}(p)$ implies $c \prec \overline{0}$, which is not allowed in this algebra. ■

With Theorems 2 and 3, and Corollary 1, we have shown that the Dijkstra algorithm with the L-lightest-TS path algebra computes loop-free, lightest paths that support hop-by-hop forwarding. In comparison with the L-lightest path algebra presented in [3], packets forwarded according to the routes computed with the L-lightest-TS algebra actually travel over the computed paths.

IV. Conclusion

We have shown that the optimality of all subpaths of paths computed by a routing function is a sufficient condition for these paths to support hop-by-hop forwarding. We have used this result to show that the L-lightest-TS path algebra, when used in a Dijkstra routing computation, results in loop-free, lightest paths that support hop-by-hop forwarding. This work builds on Sobrinho's work in [3] by showing that, while strict isotonicity is needed for loop-free hop-by-hop forwarding in the general case, there may be less restrictive solutions in circumstances where the routing algorithm itself ensures more structure for the paths actually compared during a computation. A secondary contribution is the L-lightest-TS algebra, which provides an existence proof of this claim. I would like to include a note of credit to Ted Nitz and Professor Debra Lewis from the Math Department at UCSC for their insight to use the contrapositive form of the isotonicity for proving Theorem 3.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows – Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[2] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1228–1234, September 1996.

[3] J. L. Sobrinho, "Algebra and Algorithms for QoS Path Computation and Hop-by-Hop Routing in the Internet," *IEEE/ACM Transactions on Networking*, vol. 10, no. 4, pp. 541–550, Aug. 2002.