Herding Packets: Properties Needed of Metrics for Loop-Free & Best Forwarding Paths

Bradley R. Smith
Department of Computer Engineering
University of California
Santa Cruz, CA 95064
brad@soe.ucsc.edu

Judith T. Samson
Department of Computer Engineering
University of California
Santa Cruz, CA 95064
jtsamson@soe.ucsc.edu

***Abstract*—The distributed, hop-by-hop routing architecture used in the Internet depends on algebraic properties of routing metrics to ensure traffic is forwarded over loop-free and best (LFB) paths. As the Internet evolves to serve as the converged communication infrastructure for the 21st century, the need for new metrics that violate these properties ([5], [9]) has been identified. Until recently, the behavior of routing metrics had not been studied. Recent work ([5], [9]) has presented some results identifying the requirements of metrics to ensure LFB paths, however they have not been fully characterized. Building on this work, this paper presents the necessary and sufficient conditions for routing metrics to ensure LFB paths in an Internet environment. Specifically, a metric must be strictly bounded ($a < a+b$) and monotonic ($a \leq b \Rightarrow a+c \leq b+c$) to ensure LFB forwarding paths. This paper presents the first comprehensive statement of the properties required to ensure LFB forwarding paths in the Internet.**

I. Introduction

The Internet routing architecture is based on a distributed, hop-by-hop routing model where shortest paths are computed at each node to all destinations using distance-related metrics, and a forwarding decision is made for each packet at each hop based on the packet's destination. The distributed nature of this architecture provides a more robust and scalable routing function compared with one based on centralized route computations. However, it introduces a dependency on the algebraic properties of the metrics to ensure traffic is forwarded over loop-free and best paths (we introduce the term forwarding paths to distinguish the paths traffic is actually forwarded over from the paths computed at each node).

The metrics used in the Internet have traditionally been delay or cost related, and have used well-behaved algebras composed of integer addition and comparison relations. However, as the Internet has evolved to serve as the consolidated communications infrastructure for the 21st century, the need to support new metrics and algebras has been introduced by new, time-sensitive applications involving the transmission of audio, video and telemetry data. Technology exists to implement such a choice of paths. Policy-based routing in Linux [7] and, e.g., Cisco's Virtual Routing and Forwarding (VRF) [3] support the computation of special paths for diverse performance and policy constraints, and the assignment of traffic to these different forwarding tables. However, as this paper shows, complex metrics must be used with careful attention to their algebraic properties.

These new algebras are not as well-behaved as traditional Internet routing algebras and, as has been documented elsewhere ([5], [9]), break the assumptions (implicitly) made by the Internet routing architecture. This previous work has identified sufficient properties required of these metrics to ensure traffic is forwarded over loop-free and best (LFB) forwarding paths (we use the term best to generalize the concept of shortest used for traditional, distance-based metrics). However no work has identified both the necessary and sufficient conditions for these properties.

For example, Gouda [5] identifies the property required to ensure best forwarding paths, however only identifies a property that ensures a solution for loop-free forwarding paths exists (as opposed to ensuring all locally computed solutions result in a global solution for loop-free forwarding paths). Similarly, Sobrinho [9] identifies the property required to ensure best forwarding paths, and then identifies two special cases that result in loop-free forwarding paths, but not a general property required for loop-freedom.

Building on this previous work, in this paper we develop a number of scenarios illustrating limitations of the destination-based forwarding model in forwarding traffic over LFB paths, even in the presence of LFB paths computed at each node in the network. Based on these scenarios, we identify the necessary and sufficient properties required to ensure that when traffic is forwarded along best paths computed at each hop, it will actually travel over LFB forwarding paths to its destination.

This paper is organized as follows. Section II presents definitions used in the rest of the paper. Section III reviews the scenarios we developed to give an intuition

into the seeming contradiction that forwarding of traffic along LFB paths computed at each node can actually result in traffic traveling over non-LFB forwarding paths. Section IV presents the algebraic properties we have identified as required of metrics used in such an environment to ensure LFB forwarding paths. Lastly, Section V relates these results to previous work.

II. Definitions

A *network* is defined as a pair of sets, $G = (V, E)$, where V is the set of nodes in the network and E is the set of edges that connect the nodes, and where every edge (i, j) has a weight $w(i, j)$.

A *path* p is an ordered sequence of vertices $p = (v_1, v_2, \ldots, v_{n-1}, v_n)$, each connected by an edge (v_i, v_{i+1}), from a source node $s = v_1$ to a destination node $d = v_n$ in the network. The path weight, $w(p)$, is determined by combining the weights of the edges that make up the path using addition (as specified in the path algebra described below) on the weights of the edges that compose the path (i.e. $w(p) = w(v_1, v_2) \oplus w(v_2, v_3) \oplus \ldots \oplus w(v_{n-1}, v_n)$).

In a hop-by-hop, destination-based routing environment, we make a distinction between a path and a forwarding path. In *hop-by-hop* routing [9], nodes in a network compute the shortest path to each destination, and build a forwarding table composed of the next hops on these paths for each destination. As a packet travels through the network the node at each hop independently selects the next hop, based on the forwarding table computed at that node. With *destination-based* routing a node makes a forwarding decision based only on the destination of the packet and the forwarding table described above. *Forwarding paths* are an emergent property of the collective routing tables of all nodes, are not known by any single node, and may be different from the paths computed at any given node.

In order to create best, loop-free forwarding paths, one first needs to define what "best" means, and how to compute the weight of a path. The *path algebra* provides this information. A path algebra consists of a set of metrics that describe the weight of the paths we wish to optimize, along with the operations needed to compute and compare paths. The path algebra determines how the best forwarding path through the network is found. Specifically, a path algebra is a 5-tuple $(W, \preceq, \oplus, \overline{\infty}, \overline{0})$ where:

- W is the set of weights. Edge and path weight values $(w(v_1, \ldots, v_n))$ are drawn from W.
- $\preceq$ is an total order relation on W indicating preference
  - $\preceq$ is reflexive: $a \preceq a$
  - $\preceq$ is transitive: $(a \preceq b)$ and $(b \preceq c) \Rightarrow (a \preceq c)$
  - $\preceq$ is antisymmetric: $(a \preceq b)$ and $(b \preceq a) \Rightarrow (a = b)$
  - $\preceq$ is total: either $a \preceq b$ or $b \preceq a$
- $\oplus$ is a binary function on W used to compute the weight of a path from its component paths and edges
- $\overline{0}$ and $\overline{\infty}$ are distinguished weights in W that represent the weight of the null path (i.e. path from a node to itself) and the lack of a path between two nodes, respectively. These values are never assigned to edges in a network.
  - $\overline{\infty}$ is *absorbing* for $\oplus$ $(a \oplus \overline{\infty} = \overline{\infty})$, and is a *greatest element* for $\preceq$ $(a \preceq \overline{\infty})$
- $(W, \oplus, \overline{0})$ is a *monoid*
  - $\oplus$ is associative: $a \oplus (b \oplus c) = (a \oplus b) \oplus c$
  - $\overline{0}$ is an *identity element* for $\oplus$: $a \oplus \overline{0} = a$

Lastly, a path algebra is said to be *optimal* if, for the best path $p = (v_0, \ldots, v_n)$, all subpaths of p are also best paths.

III. Examples

In this section we review a number of path algebras that illustrate the range of behavior possible with respect to the loop-free and best-path properties of the forwarding paths resulting from the different algebras.

Figures 1-5 illustrate the best path from all nodes to node D using different path algebras. Figure 1 shows the best paths for the *Shortest* path algebra where weights are composed of an integer distance value, $\oplus$ is defined as integer addition ("+"), and $\preceq$ is defined as integer $\leq$. Notice that if we trace the path taken by a packet sent from A to D, the forwarding decision made at each hop follows the best path computed at A. Specifically A, following the path (A, B, C, D) it computed, forwards a packet to B; similarly, B (following its path (B, C, D)) forwards the packet to C, and C forwards it to D. Therefore, packets sent from A follow a LFB forwarding path to D.

Figure 2 shows best paths defined by the *Widest-Shortest* metric where weights are composed of the integer pair $(delay, bandwidth)$, $(d_1, b_1) \oplus (d_2, b_2)$ is defined as $(d_1 + d_2, Min(b_1, b_2))$, and $(d_1, b_1) \preceq (d_2, b_2)$ as $(d_1 < d_2)$ *or* $((d_1 = d_2)$ *and* $(b_1 \geq b_2))$ (intuitively, prefer shortest paths and, among all shortest paths, prefer those with maximum bandwidth). Following the path taken by a packet sent from A to D, we see that the forwarding decision made at each hop follows a path with the same weight as the best path computed at A. Note that, from $A's$ perspective, there are two equally best paths to D ((A, B, D) and (A, B, C, D), both with weight $(3, 5)$), while from $B's$ perspective there is only one $(w(B, C, D) = (2, 10))$, which is clearly better than $w(B, D) = (2, 5) \succ (2, 10)$. The important observation

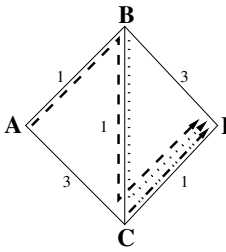

Fig. 1. Shortest

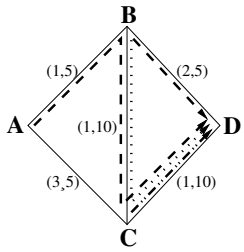

Fig. 2. Widest-Shortest

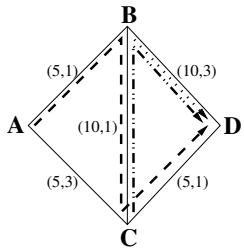

Fig. 3. Shortest-Widest

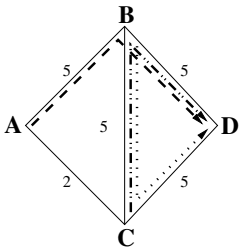

Fig. 4. Widest

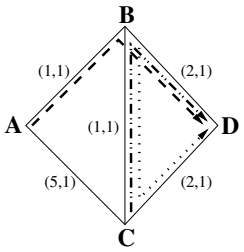

Fig. 5. "Slope"

being that the set of best paths from $B's$ perspective is a subset of the best paths from $A's$ perspective, thereby ensuring that traffic follows LFB forwarding paths.

Figure 3 shows best paths defined by the *Shortest-Widest* metric where weights are composed of the integer pair $(bandwidth, delay)$, $(b_1, d_1) \oplus (b_2, d_2)$ is defined as $(Min(b_1, b_2), d_1 + d_2)$, and $(b_1, d_1) \preceq (b_2, d_2)$ as $(b_1 > b_2)$ *or* $((b_1 = b_2)$ *and* $(d_1 \leq d_2))$ (prefer paths with the most bandwidth and, among all maximum bandwidth paths, prefer the shortest ones). Following the path taken by a packet sent from A to D we see that the forwarding decision at node B selects a different next hop (D) from the next hop on the shortest path from A (C), resulting in the packet being forwarded over a path that is not the shortest path from A to D. Therefore hop-by-hop, distributed routing using the *Shortest-Widest* metric cannot guarantee that packets will follow shortest paths to their destination.

Figure 4 shows a possible solution for best paths defined by the *Widest* metric where weights are composed of an integer weight value (which could represent e.g. bandwidth), $b_1 \oplus b_2$ is defined as $Min(b_1, b_2)$, and $b_1 \preceq b2$ is defined as $b_1 \geq b_2$ (i.e. prefer paths with the most bandwidth). Following the path taken by a packet from A to D we see that a loop is entered between B and C. Note that the actual paths computed at the different nodes to traverse the links connecting B, C, and D depend on the implementation of the the shortest path computation. The paths shown are possible paths, and other path selections are possible that would not result in the loop. The important point is that it is not possible to ensure LFB forwarding paths with destination-based, hop-by-hop routing using the *Widest* metric.

Lastly, Figure 5 shows the best paths defined by what we call the *Slope* metric where weights are composed of the integer pair $(distance, cost)$, $(d_1, c_1) \oplus (d_2, c_2)$ is defined as $(d_1 + d_2, c_1 + c_2)$, and $(d_1, c_1) \preceq (d_2, c_2)$ as $(d_1/c_1) \leq (d_2/c_2)$ (i.e. prefer paths with a lesser slope, or cost per unit of distance). Similar to the *Widest* metric, following the path taken from A to D, a loop is entered between B and C. Note, in contrast to *Widest*, with the *Slope* metric there are no choices in the selection of these paths, and the path costs increase as traffic is forwarded in the loop.

These examples show a progression of increasingly misbehaving metrics that lead to progressively deteriorating forwarding behavior. In the next section we present the properties needed to ensure LFB forwarding paths, and in Section 5 explain the behavior of the example metrics in terms of these properties.

IV. Herding Properties

The properties needed to ensure LFB forwarding paths are *monotonicity* and *strict boundedness*. These are derived from the properties presented by Gouda [5] with the same names. Sobrinho [9] uses the term *isotonicity* for what we call monotonicity.

Specifically, given a path algebra $(W, \preceq, \oplus, \overline{\infty}, \overline{0})$, for all $a, b, c \in W$: *monoticity* is the property that $a \preceq b \Rightarrow a \oplus c \preceq b \oplus c$, and *boundedness* is the property that $a \preceq a \oplus b$. Both properties have strict versions; i.e. a path algebra is: *strictly monotonic* if $a \prec b \Rightarrow a \oplus c \prec b \oplus c$, and *strictly bounded* if $a \prec a \oplus b$.

Note that both versions of monotonicity infer the corresponding version of boundedness. This can be seen by setting a to $\overline{0}$ in the definition. I.e., for monotonicity with $a = \overline{0}$, $\overline{0} \preceq b \Rightarrow \overline{0} \oplus c = c \preceq b \oplus c$; since $\overline{0} \preceq b$ is always true (we assume no edge is ever assigned a weight of $\overline{0}$), the implication that $c \preceq b \oplus c$ must also always be true. A similar argument can be made for the strict versions of the properties.

Intuitively, monotonicity is the property that extending two paths with the same edge will not reverse their order, and strict monotonicity is the property that extending two paths with the same edge retains their ordering. Similarly, boundedness is the property that a path does not get better as it is extended, and strict boundedness that a path only gets worse as it is extended. Using these descriptions, it is also intuitive that a path algebra that is monotonic must also be bounded (if extending two paths with the same edge does not reverse their order, then the same must also be true when one of the starting paths is null, i.e. with a weight of $\overline{0}$), and similarly for the strict properties.

An interesting observation, which lies at the core of the insight to the properties required for LFB forwarding

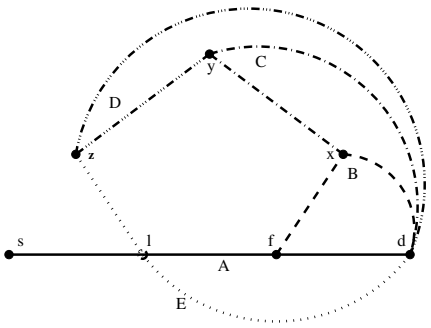

Fig. 6. strictly-bounded $\Rightarrow$ loop-free

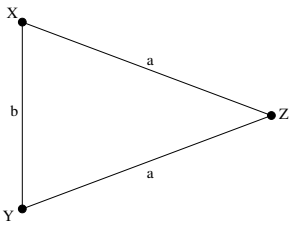

Fig. 7. loop-free $\Rightarrow$ strictly-bounded

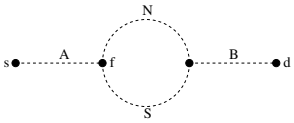

Fig. 8. monotonicity and best forwarding paths

paths, is the fact that a path is monotonic does not imply that it is strictly bounded. In other words, the fact that a path only gets worse as it is extended (strictly bounded) is an additional constraint on an algebra beyond the fact that extending two paths with the same edge will not reverse their order (monotonic). Section V shows how all combinations of these properties are demonstrated in the example path algebras from Section III.

The remainder of this section presents proofs that strictly bounded path algebras always result in loop-free forwarding paths, that monotonic path algebras always result in best forwarding paths, and that strictly monotonic (and, by implication, strictly bounded) path algebras always produce optimal forwarding paths.

A. Strictly Bounded and Loop-Free

First we show that strictly bounded is the defining property of path algebras that guarantee loop-free forwarding paths. We prove this in two parts presented in the following Lemmas.

Lemma 1. *If a path algebra is strictly bounded then it guarantees loop free forwarding paths.*

Proof: By contradiction. Assume that a path algebra $(W, \preceq, \oplus, \overline{\infty}, \overline{0})$ is strictly bounded, but that there are best paths computed at different nodes using the algebra that result in forwarding paths that loop. With no loss of generality (i.e. this illustration can be generalized to a loop of any number of nodes > 1), Figure 6 illustrates this scenario for a 5 node loop. Each segment connecting two nodes in this graph represents a subpath containing 0 or more nodes, with no branches occurring in the subpath (i.e. at each node in the subpath prior to the fork, the path computed at that node is the concatenation of the link to its successor and the path computed at the successor). The weight of path $(s, \ldots, l, \ldots, f, \ldots, d)$ is called A, that of path $(f, \ldots, x, \ldots, d)$ is called B, and similarly for paths C through E.

At each fork in a path (e.g. node f on path A) the forked path taken by that node (e.g. path B taken by node f) must be no worse than the portion remaining at the fork of the path computed at the source that led to the fork (i.e. $B \preceq A_{f,d}$, where $A_{f,d}$ is the weight of the portion of path A between nodes f and d). This leads to the sequence of inequalities shown in Table I:

The last inequality is a contradiction to the assumption that the path algebra is strictly bounded. This proof is similar to one given by Garcia-Luna [4]. □

Lemma 2. *If forwarding paths computed using a path algebra are guaranteed to be loop-free then the path algebra is strictly bounded.*

Proof: By contradiction. Assume we have a path algebra $(W, \preceq, \oplus, \overline{\infty}, \overline{0})$ that is not strictly bounded, but always results in loop-free forwarding paths. Therefore, there exist an $a, b \in W$ where $a \succeq a \oplus b$. Using these elements we can construct the graph shown in Figure 7.

Given this graph, if $a \succ a \oplus b$, traffic from Y to Z will take the path (Y, X, Z), and traffic from Z to Y will take path (X, Y, Z), resulting in a loop. On the other hand, if $a = a \oplus b$, while traffic between Y and Z *may* follow loop free paths, this is not guaranteed (i.e., depending on the vagaries of the implementation of the shortest path computation, the computations could result in a loop). Both cases violate the assumption that paths computed using the path algebra are guaranteed to result in loop-free forwarding paths. □

With these two Lemmas we can now prove the following theorem.

Theorem 1. *A path algebra guarantees loop free forwarding paths if and only if it is strictly bounded.*

Proof: By Lemma 1 we have that *if a strictly bounded path algebra is used in a destination-based, hop-by-hop routing computation, then the set of routes computed at the nodes in the network are guaranteed to result in loop free forwarding paths*, and by Lemma 2 we have that *if the path algebra used in a destination-based, hop-by-hop routing computation is guaranteed to result in loop free forwarding paths, then the path algebra must be strictly bounded.* □

$B \preceq A_{f,d}$	Assumption of fork at f.
$C \preceq B_{x,d} \prec B = w(f,\ldots,x) \oplus B_{x,d}$	Assumption of fork at x and strictly bounded.
$D \preceq C_{y,d} \prec C = w(x,\ldots,y) \oplus C_{y,d}$	Assumption of fork at y and strictly bounded.
$E \preceq D_{z,d} \prec D = w(y,\ldots,z) \oplus D_{z,d}$	Assumption of fork at z and strictly bounded.
$A_{l,d} \preceq E_{l,d} \prec E = w(z,\ldots,l) \oplus E_{l,d}$	Given $l's$ next hop is on path A and strictly bounded.
$A_{l,d} = w(l,\ldots,f) \oplus A_{f,d} \prec A_{f,d}$	Transitivity of $\prec$. $\Rightarrow\Leftarrow$

TABLE I
INEQUALITIES LEADING TO LOOP-FREEDOM.

B. Monotonicy and Best

Next we show that monotonicity is the defining property of path algebras that guarantee best forwarding paths. We prove this in two parts presented in the following Lemmas.

Lemma 3. *If a path algebra is monotonic then traffic is guaranteed to follow best forwarding paths.*

Proof: By contradiction. In Figure 8, assume the algebra is monotonic, the path (A, N, B) is the best path, but traffic is forwarded over path (A, S, B). We know that traffic is forwarded over best path (A) to a point f, where the paths diverge. The fact that f forwards via (S, B) implies that $w(S, B) \prec w(N, B)$ and therefore, by monotonicity, $w(A) \oplus w(S, B) \preceq w(A) \oplus w(N, B)$. However, this contradicts the assumption that (A, N, B) is best (i.e. that $w(A) \oplus w(N, B) \prec w(A) \oplus w(S, B)$). □

Lemma 4. *If a given path algebra always results in best forwarding paths then it is monotonic.*

Proof: By contradiction. Assume use of the path algebra always results in best forwarding paths, but that the path algebra is not monotonic. This implies there exist three weights $a, b, c \in W$ where $a \preceq b$ however $a \oplus c \succ b \oplus c$. Using Figure 8 again, we can construct a scenario where $a = w(N, B), b = w(S, B), c = w(A)$ and $a \prec b$, however $a \oplus c \succ b \oplus c$ (e.g. see *Shortest-Widest* in Section 3). f will forward traffic from s to d over path (N, B) even though, from $A's$ point of view, path (A, S, B) is better than path (A, N, B) (i.e. $w(A, S, B) = b \oplus c \prec a \oplus c = w(A, N, B)$). This contradicts the assumption that the path algebra always results in best forwarding paths. □

With these two Lemmas we can now prove the following theorem.

Theorem 2. *A path algebra guarantees best forwarding paths if and only if it is monotonic.*

Proof: By Lemma 3 we have that *if a monotonic path algebra is used in a destination-based, hop-by-hop routing computation, then the set of routes computed at the nodes in the network is guaranteed to result in best forwarding paths*, and by Lemma 4 we have that *if the path algebra used in a destination-based, hop-by-hop routing computation is guaranteed to result in best forwarding paths, then the path algebra must be monotonic.* □

C. Strict Monotonicity and Optimal

Lastly, we will show that strict monotonicity is the defining property of an optimal path algebra. We prove this in two parts, presented in the following Lemmas.

Lemma 5. *If a path algebra is strictly monotonic then traffic is guaranteed to follow optimal forwarding paths.*

Proof: By contradiction. Assume a path algebra is strictly monotonic, but results in forwarding paths that are not optimal. Using Figure 8 to illustrate, given the path algebra is not optimal, we can construct a case where both paths from s to d are best (i.e. $w(A, N, B) = w(A, S, B)$), but only (S, B) is best from f (i.e. $w(S, B) \prec w(N, B)$). However, since the path algebra is strictly monotonic, $w(S, B) \prec w(N, B) \Rightarrow w(S, B) \oplus w(A) \prec w(N, B) \oplus w(A)$, which contradicts the assumption of non-optimality. □

Lemma 6. *If a given path algebra always results in optimal forwarding paths then it is strictly monotonic.*

Proof: By contradiction. Assume a path algebra always results in optimal forwarding paths, but is not strictly monotonic. Since the path algebra is not strictly monotonic there exist $a, b, c \in W$ where $a \prec b$ but $a \oplus c \succeq b \oplus c$. Using these values we can construct a case, again using Figure 8, where $w(N, B) = a, w(S, B) = b$ and $w(A) = c$. Given the assumptions of optimality and $w(N, B) = a \prec b = w(S, B)$ we can infer (A, N, B) is a best path from s to d. However, since $a \oplus c \succeq b \oplus c$, it must also be true that (A, S, B) is a best path from s to d and, given $a \prec b$, (A, S, B) is not optimal, which contradicts the assumption of optimality. □

We can now prove the following theorem.

Theorem 3. *A path algebra is strictly monotonic if and only if it results in optimal forwarding paths.*

Proof: By Lemma 5 we have that *if a strictly monotonic path algebra is used in a destination-based, hop-by-hop routing computation, then the set of routes computed at the nodes in the network is guaranteed to*

result in optimal forwarding paths, and by Lemma 6 that *if the path algebra used in a destination-based, hop-by-hop routing computation is guaranteed to result in optimal forwarding paths, then the path algebra must be strictly monotonic.* □

In the following section we use these theorems to explain the behavior of the examples presented in Section III.

V. Interpreting the Examples

Section III presented a series of examples that demonstrate a progression of increasingly misbehaving metrics that lead to progressively deteriorating forwarding behavior. In this section we explain the behavior of these metrics using the properties presented in Section IV. First we include the following definition of $\leq$ for $\mathbb{N}_0$ (the natural numbers, including 0):

Definition 1. For all $a, b \in \mathbb{N}_0,\ a \leq b$ if and only if there exists some $c \in \mathbb{N}_0$ such that $a + c = b$.

The remainder of this section identifies the properties each example path algebra has, and how these properties explain the behavior illustrated in the examples from Section III.

Theorem 4. *The* Shortest *path algebra is strictly monotonic.*

Proof: If $a < b$ then, by the definition of $\leq$, there must be an $x \in W$ where $a + x = b$. Similarly, for $a + c < b + c$ to be true, there must be a $y \in W$ where $(a + c) + y = (b + c)$. Setting $y = x$ satisfies this equality. □

Therefore the *Shortest* path algebra is optimal and is guaranteed to result in LFB forwarding paths. Furthermore, the distinguishing characteristic of optimality is that $A's$ view of it's path to D will be consistent with the view of all intermediate hops; therefore a circumstance such as illustrated in Figure 2 for *Widest-Shortest* where one possible shortest path computed at A (e.g. (A, B, D)) is actually not shortest at B, will not occur when routes are computed using *Shortest*.

Theorem 5. *The* Widest-Shortest *path algebra is monotonic* ($(d_a, b_a) \preceq (d_b, b_b) \Rightarrow (d_a, b_a) \oplus (d_c, b_c) \preceq (d_b, b_b) \oplus (d_c, b_c)$) *and strictly bounded* ($(d_a, b_a) \prec (d_a, b_a) \oplus (d_b, b_b)$).

Proof: First we show it is monotonic. Given $(d_a, b_a) \preceq (d_b, b_b)$ we know either $d_a < d_b$ or $d_a = d_b$. Similar to what we showed for the $\mathbb{N}_0$ values in *Shortest*, whatever relationship exists between d_a and d_b will hold for the values $(d_a + d_c)$ and $(d_b + d_c)$. If $d_a < d_b$ then $(d_a + d_c) < (d_b + d_c)$, $(d_a, b_a) \oplus (d_c, b_c) \preceq (d_b, b_b) \oplus (d_c, b_c)$ is true, and the the monotonic property holds. If, however, $d_a = d_b$ (and therefore $(d_a + d_c) = (d_b + d_c)$), then $b_a \geq b_b$ and it is not possible that $Min(b_a, b_c) < Min(b_b, b_c)$; therefore $Min(b_a, b_c) \geq Min(b_b, b_c)$, the monotonic property still holds, and *Widest-Shortest* is monotonic.

Next we show it is strictly bounded. Since $d_b \neq 0$, it must be true (based on the definition of for $\mathbb{N}_0$ given above) that $d_a < d_a + d_b$, therefore $(d_a, b_a) \prec (d_a, b_a) \oplus (d_b, b_b)$ must also be true, and *Widest-Shortest* is strictly bounded. □

Therefore the Widest-Shortest path algebra guarantees LFB, but not optimal paths. This can be seen in the example illustrated in Figure 2 where traffic is forwarded over a LFB path, though not all paths computed at node A are optimal (e.g. the path (A, B, D) appears best to A, however it is not best for B).

Theorem 6. *The* Shortest-Widest *path algebra is not monotonic, but it is strictly bounded.*

Proof: The following example illustrates that *Shortest-Widest* is not monotonic: $(10, 3) \preceq (5, 1)$ however $(10, 3) \oplus (5, 1) = (5, 4) \succ (5, 2) = (5, 1) \oplus (5, 1)$. We now show it is strictly bounded ($(b_a, d_a) \prec (b_a, d_a) \oplus (b_b, d_b)$). Since it cannot be true that $b_a < Min(b_a, b_b)$, either $b_a > Min(b_a, b_b)$ or $b_a = Min(b_a, b_b)$. If $b_a > Min(b_a, b_b)$ then the *Strictly-Bounded* property holds. If, however, $b_a = Min(b_a, b_b)$ then the remaining test is $d_a < d_a + d_b$, which (based on Definition 1 of $\leq$ for $\mathbb{N}_0$) must be true, and *Shortest-Widest* is strictly-bounded. □

Therefore, the Shortest-Widest path algebra guarantees loop-free but not best forwarding paths. This can be seen in the example illustrated in Figure 3 where traffic is forwarded over a loop-free but not best path (i.e. traffic from A to D is forwarded over path (A, B, D), which is loop free, but worse, from $A's$ perspective, than path (A, B, C, D)).

Theorem 7. *The* Widest *path algebra is not strictly-bounded, but it is monotonic.*

Proof: An example of non-strictly bounded behavior is $5 = 5 \oplus 10$. Monotonicity ($b_a \preceq b_b \Rightarrow (b_a \oplus b_c) \preceq (b_b \oplus b_c)$) is shown as follows. Since $b_a \preceq b_b \Rightarrow b_a \geq b_b$, it is not possible for $Min(b_a, b_c) < Min(b_b, b_c)$. Therefore it must be true that $Min(b_a, b_c) \geq Min(b_b, b_c)$, and $Min(b_a, b_c) \preceq Min(b_b, b_c)$. □

Therefore, the Widest path algebra guarantees best but not loop-free forwarding paths. This is a strange, and perhaps not very useful concept. However it can be seen in the example illustrated in Figure 4 where traffic is forwarded in a loop but the cost of the path traveled in the loop is never "worse" than the "best" path (if we do not include actually arriving at the destination in the definition of "best"!).

Lastly, the *Slope* path algebra is neither monotonic

(e.g. $(10,10) = 10/10 = 1 < 2 = 2/1 = (2,1)$, but $(10,10) \oplus (1,4) = 11/14 \approx .8 > .6 = 3/5 = (2,1) \oplus (1,4)$), nor strictly bounded (e.g. $(1,1) = 1 > 2/3 = (2,3) = (1,1) \oplus (1,2)$). Therefore the *Slope* algebra guarantees neither loop free nor best paths. This degenerate behavior can be seen in the example illustrated in Figure 5 where traffic is forward in a loop and the cost of the path traveled gets progressively worse as the traffic is forwarded around the loop.

This section has shown that the monotonic and strictly bounded properties identified in Section IV fully explain the behavior of the examples from Section III. In the following section we review previous work and show how the findings from these works are explained and extended by the results of Section IV.

VI. Previous Work

Foundational work was done in this area by Jaffe [6] where he presented and analyzed algorithms for computing routes that satisfied constraints on two additive metrics (he used weight and length). Wang and Crowcroft [10] were the first to present a solution for computing routes in the context of a concave (or minmax, such as bandwidth) and an additive metric. Cavendish and Gerla [1] presented a modified Bellman-Ford algorithm with complexity of $O(n^3)$, which computes multi-constrained paths if all metrics of paths in a network are either non-decreasing or non-increasing as a function of the hop count. Several other solutions have been proposed by Chen, Jaffe, and Van Mieghem [2], [6], [8] for computing approximate solutions to the QoS routing problem based on mapping metrics to a reduced range, or using a function of the metrics for routing.

More recently, work has been done by Gouda [5] and Sobrinho [9] on the implications of these new metrics on the routing computation and, to a more limited degree, on the forwarding process. These works are most relevant to this paper. Gouda [5] presents a theory of metrics for computing LFB paths in a network, which are called maximizable routing metrics. The monotonic and bounded properties are defined, and shown to be necessary and sufficient for the existence of an in-tree of LFB paths (called a maximal metric tree) rooted at each node. This work does not address the question of what properties are required to ensure all possible distributed computations of these paths result in maximal metric trees, and thus guarantee LFB forwarding paths as described here. Our work extends that of Gouda[5] to answer these questions.

Sobrinho [9] defines an algebra of metrics for use in computing routes in a network, and identifies the monotonic property (called isotonic in the paper) as necessary and sufficient properties of the algebra to ensure a Dijkstra shortest-path computation yields LFB paths, and that the resulting forwarding paths are best. It further shows that without strict monotonicity the forwarding paths may loop, and then presents a lexicographic ordering based on the underlying metric that ensures LFB forwarding paths (assuming the underlying metric is monotonic). In terms of our work, Sobrinho [9] two special cases of metrics that are strictly bounded: strictly monotonic metrics (recall by Theorem 3 that strictly monotonic path algebras guarantee optimal forwarding paths), and lexicographic lightness (which can be shown to be strictly bounded and therefore, by Theorem 1, to guarantee loop freedom). Our work extends that of Sobrinho [9] by identifying the defining property for loop freedom (strictly bounded). Lastly, Yang [11] presents similar work focusing on forwarding paths, however the paper identifies properties for optimal paths (in our terminology) compared with our focus on LFB forwarding paths.

VII. Conclusion

In this paper we show that destination-based, hop-by-hop routing is surprisingly delicate with respect to the algebraic properties of the metrics used to compute routes at each hop, and that routing metrics must be relatively well-behaved to ensure that traffic actually travels over LFB forwarding paths. Specifically, for a metric to ensure LFB forwarding paths it must be strictly bounded ($a \prec a \oplus b$) and monotonic ($a \preceq b \Rightarrow a \oplus c \preceq b \oplus c$).

These findings identify serious limitations to metrics used for routing in the Internet.

References

[1] Dirceu Cavendish and Mario Gerla. Internet QoS Routing Using the Bellman-Ford Algorithm. In *Proceedings IFIP Conference on High Performance Networking*, 1998.

[2] Shigang Chen and Klara Nahrstedt. An Overview of Quality of Service Routing for Next-Generation High-Speed Networks: Problems and Solutions. *IEEE Network*, pages 64–79, Nov 1998.

[3] Cisco. *MPLS VPN: VRF Selection using Policy Based Routing*, Aug 2004.

[4] J.J. Garcia-Luna-Aceves. Loop-Free Routing Using Diffusing Computations. *IEEE/ACM Transactions on Networking*, 1(1):130–141, Feb 1993.

[5] Mohamed G. Gouda and Marco Schneider. Maximizable Routing Metrics. *IEEE/ACM Transactions on Networking*, 11(4):663–675, Aug 2003.

[6] Jeffrey M. Jaffe. Algorithms for Finding Paths with Multiple Constraints. *Networks*, 14(1):95–116, 1984.

[7] Matthew G. Marsh. *Policy Routing With Linux*, March 2001.

[8] Piet Van Mieghem, Hans De Neve, and Fernando Kuipers. Hop-by-hop Quality of Service Routing. *Computer Networks*, 37:407–423, Nov 2001.

[9] João Luís Sobrinho. Algebra and Algorithms for QoS Path Computation and Hop-by-Hop Routing in the Internet. *IEEE/ACM Transactions on Networking*, 10(4):541–550, Aug 2002.

[10] Zheng Wang and Jon Crowcroft. Quality-of-Service Routing for Supporting Multimedia Applications. *IEEE Journal of Selected Areas in Communications*, 14(7):1228–1234, Sep 1996.

[11] Yaling Yang and Jun Wang. Design Guidelines for Routing Metrics in Multihop Wireless Networks. In *Proceedings IEEE INFOCOM 2008*, pages 2288–2296. IEEE, 2008.