Fundamental Differences Among Vectoring Routing Protocols on Non-Isotonic Metrics

João Luís Sobrinho, *Senior Member, IEEE*

***Abstract*—All of RIP, EIGRP, BGP, DSDV, and Babel have been classified as distance-vector protocols. The common denomination belies their different outcomes when routing on metrics that do not satisfy the algebraic property of left-isotonicity. Under this circumstance, we show that non-restarting vectoring protocols, such as RIP, EIGRP, and BGP, route correctly, along paths that can be characterized as the best possible subject to a destination-based forwarding strategy; whereas, contrastingly, restarting vectoring protocols, such as DSDV and a mode of operation of Babel, do not route correctly.**

***Index Terms*—Routing, routing protocols, optimal paths, destination-based forwarding, RIP, BGP, EIGRP, DSDV, Babel.**

I. Overview

THERE is a useful level of abstraction on the basis of which all routing protocols can be described in common terms. Letting the notion of *attribute* stand for all metrics of a path that are relevant in a given context—such as hop-count, capacity, delay, available bandwidth, loss probability, energy consumption, co-channel interference, etc. [1]–[3]—routing protocols iterate *election* and *extension* operations on attributes. An election operation produces the best of two attributes, thereby ranking paths. An extension operation composes two attributes into a third one, modeling how the attribute of a path is obtained from the attributes of concatenated sub-paths. At this level of abstraction, routing protocols differ on how election and extension operations are interleaved in time and space across the nodes of a network. The insight was offered several years ago that the relationship between the local actions of a routing protocol and the global behaviors induced by them is determined by algebraic properties entwining election and extension [4].

Vectoring protocols generalize distance-vector protocols [5]. These protocols instantiate a separate computation process per destination. For a given destination, nodes extend attributes advertised by their out-neighbors into candidate attributes, elect an attribute from among the candidates, and advertise it to in-neighbors. RIP [6], EIGRP [7], BGP [8], DSDV [9], and Babel [10] belong to this class of protocols, despite the many differences in implementation—such as routing state, use of timers, link layer assumptions, etc. An algebraic property with a strong impact on the behavior of these protocols is left-isotonicity, which means that the relative preference between two attributes is preserved between the extensions of a third attribute with each of them. If left-isotonicity holds, then the stable state of vectoring protocols guides data-packets along optimal paths. A first contribution of this letter is the identification of two subclasses of vectoring protocols that exhibit strikingly different behaviors in the absence of left-isotonicity. They will be called *non-restarting* and *restarting* vectoring protocols.

In non-restarting vectoring protocols, the destination initiates the routing computation only once. The elected attribute at a node may improve or worsen as a function of events in the network, such as the introduction of a new link or the failure of a link. The election of an attribute at a node that is worse than the one previously elected may lead to the exploration of ever worse attributes, a condition known as count-to-infinity [5], and to long-lived forwarding loops. RIP, EIGRP, and BGP are examples of non-restarting vectoring protocols. RIP is the simplest of them. It operates on a small set of attributes, thus guaranteeing a stop to count-to-infinity. BGP has advertisements contain the full path traversed by them away from the destination and invalidates paths that loop back to the same node. EIGRP activates a coordination process among nodes allowing updates to worse attributes without triggering count-to-infinity or creating forwarding loops. In the absence of left-isotonicity, non-restarting vectoring protocols guide data-packets along paths that have been qualified with the non-descriptive terms sub-optimal or local-optimal [4]. A second contribution of this letter is a characterization of the paths found by non-restarting vectoring protocols as the best possible under the constraint of destination-based forwarding of data-packets.

In restarting vectoring protocols, the destination continually initiates independent routing computation instances, with attributes elected during the latest instance superseding those elected in earlier instances. During a computation instance, elected attributes are only allowed to improve, a condition that is sufficient to prevent count-to-infinity and forwarding loops.[1] If events in the network require a node to worsen its elected attribute, then the node waits for the next computation instance in order to elect a new attribute. In the meantime, the node is black-holed for the destination, meaning that it discards data-packets intended there. DSDV is the prototypical restarting vectoring protocol. In DSDV, the destination initiates a new routing computation instance periodically. Babel borrows from DSDV the idea of iterated computation instances and allows attributes from the most recent instance to be preferred for election, in which case it too behaves as a restarting vectoring protocol. In Babel, a black-holed node explicitly requests the initiation of a new computation instance from the destination.

This work was supported in part by Fundação para a Ciência e Tecnologia under Project UID/EEA/50008/2019.

J. L. Sobrinho is with Instituto Superior Técnico, Universidade de Lisboa, and Instituto de Telecomunicações, Portugal (e-mail: joao.sobrinho@lx.it.pt).

[1]There are special cases where a node could worsen its elected attribute knowing that a forwarding loop would not be created [11]. These special cases do not detract from the conclusions of this letter and are not considered.

The request may be guided by the elected attributes of previous computation instances, or by flooding, or by a mixture of the two. A third contribution of this letter is to show that, in the absence of left-isotonicity, restarting vectoring protocols can leave a node black-holed forever and, thus, cannot be counted upon to deliver data-packets.

Section II reviews the basics of the algebraic framework for routing protocols. Through an example, Section III discusses the behavior of non-restarting vectoring protocols, while Section IV shows that restarting vectoring protocols fail to route correctly if left-isotonicity does not hold. Section V elaborates on the implications of our conclusions. The Appendix proves that non-restarting vectoring protocols guide data-packets on optimal paths constrained by destination-based forwarding.

II. Routing algebra and left-isotonicity

A *routing algebra* is a triple $(S, \sqcap, \oplus)$ composed of a set S of attributes, a binary election operation on attributes, $\sqcap$, and a binary extension operation on attributes, $\oplus$. Operation $\sqcap$ is selective, associative, commutative, and has an identity. Operation $\oplus$ is associative and also has an identity. The election of attributes may equivalently be modeled by a total order of preferences $\preceq$ such that $a \preceq b$ if $a \sqcap b = a$. The total order of preferences and extension are entwined by *inflation*:

$$\forall_{a,b \in S}\ b \preceq a \oplus b \text{ and } b \preceq b \oplus a,$$

which is interpreted as stating that the attribute of a path is not preferred to the attribute of any of its sub-paths. An attribute $a \in S$ is *strictly-left-inflationary* if

$$\forall_{b \in S}\ b \prec a \oplus b.$$

We assume that the attribute of every cycle in every network is strictly-left-inflationary. The algebraic properties above typically hold on metrics used in practice.

A key algebraic property possessed only by some metrics is *left-isotonicity*:

$$\forall_{a,b,c \in S}\ b \preceq c \text{ implies } a \oplus b \preceq a \oplus c.$$

The prototypical non-isotone routing algebra used in this letter is the shortest-widest path algebra $(S_{\text{WL}}, \sqcap_{\text{WL}}, \oplus_{\text{WL}})$. Set S_{WL} is composed of width-length pairs, with width positive or infinity and length nonnegative, plus the special pair with zero width and infinite length that represents unreachability:

$$S_{\text{WL}} = (\mathbb{R}^+ \cup \{+\infty\}) \times \mathbb{R}_0^+ \cup \{(0, +\infty)\}.$$

Widths extend with minimum and lengths with addition. For all $(w, l), (w', l') \in S_{\text{WL}}$,

$$(w, l) \oplus_{\text{WL}} (w', l') = (\min\{w, w'\}, l + l'),$$

with $(+\infty, 0)$ the identity of $\oplus_{\text{WL}}$. Width-lengths with greater widths are preferred; in case of a tie among widths, smaller lengths are preferred. For all $(w, l), (w', l') \in S_{\text{WL}}$,

$$(w, l) \sqcap_{\text{WL}} (w', l') = \begin{cases} (w, l), & \text{if } w > w' \vee (w = w' \wedge l \leq l'); \\ (w', l'), & \text{otherwise,} \end{cases}$$

with $(0, +\infty)$ the identity of $\sqcap_{\text{WL}}$.

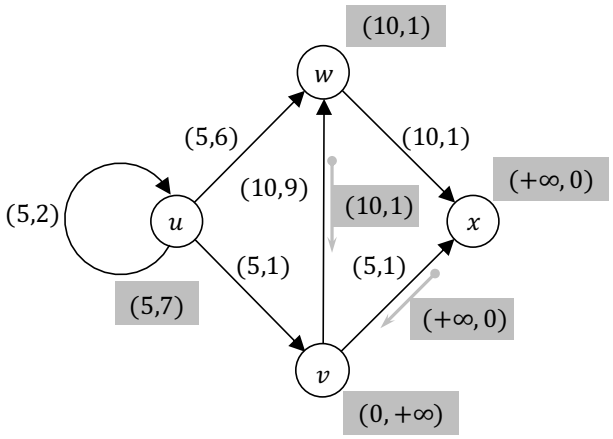

Fig. 1. Each link is annotated with a width-length. Grey rectangles and arrows indicate the routing state at a moment in time discussed in Sections III and IV. A non-restarting vectoring protocol routes data-packets to x along paths uwx and vwx, whereas a restarting vectoring protocol may leave u black-holed forever.

That the shortest-widest path algebra is not left-isotone can be verified with the trio of width-lengths $(5, 1)$, $(10, 10)$, and $(5, 1)$. We have $(10, 10) \prec_{\text{WL}} (5, 1)$, but $(5, 1) \oplus_{\text{WL}} (10, 10) = (5, 11) \succ_{\text{WL}} (5, 2) = (5, 1) \oplus_{\text{WL}} (5, 1)$.

III. Non-restarting vectoring protocols

The network of Figure 1 is used to illustrate the behavior of vectoring protocols when left-isotonicity does not hold. The self-loop at u is a synthetic representation of a cycle containing u. Each link is annotated with its width-length and the shortest-widest path algebra is assumed. The width-lengths of link uv, path vwx, and link vx, respectively, $(5, 1)$, $(10, 10)$, and $(5, 1)$, form the trio of width-lengths violating left-isotonicity presented in the previous section.

Destination x initiates the routing computation by advertising $(+\infty, 0)$ to w and v. Upon receiving the advertisement from x, w extends it to candidate width-length $(10, 1) = (10, 1) \oplus_{\text{WL}} (+\infty, 0)$, and elects and advertises this width-length to u and v. Node u receives advertisement $(10, 1)$ from w, and elects and advertises width-length $(5, 7) = (5, 6) \oplus_{\text{WL}} (10, 1)$ around the self-loop. The latter advertisement is received back at u to no further consequence there. The state of the protocol at this moment in time is marked in the figure. There are two advertisements in transit: $(+\infty, 0)$ from x to v and $(10, 1)$ from w to v. We distinguish two cases.

Case 1: advertisement $(10, 1)$, sent by w, arrives first at v. Then, v elects $(10, 10) = (10, 9) \oplus_{\text{WL}} (10, 1)$ and advertises this width-length to u. Later, when the advertisement $(+\infty, 0)$ arrives at v from x, v extends it to candidate width-length $(5, 1) = (5, 1) \oplus_{\text{WL}} (+\infty, 0)$, which is less preferred than $(10, 10)$ and, thus, is not elected. When the advertisement $(10, 10)$ is received at u from v, u extends it to candidate width-length $(5, 11) = (5, 1) \oplus_{\text{WL}} (10, 10)$, which is less preferred than $(5, 7)$ and also not elected. A stable state has been reached with u electing $(5, 7)$, which is the width-length of path uwx.

This was a fortunate interleaving of advertisements in that the first elected width-length at every node is also the

final one. However, the stable state does not guide data-packets along shortest-widest paths. Data-packets originated at u travel to x along path uwx, width-length $(5, 7)$, whereas the unique shortest-widest path from u to x is path uvx, width-length $(5, 2)$.

Case 2: advertisement $(+\infty, 0)$, sent by x, arrives first at v. Then, v elects $(5, 1) = (5, 1) \oplus_{\text{WL}} (+\infty, 0)$ and advertises this width-length to u. When advertisement $(10, 1)$ arrives at v from w, v extends it to candidate width-length $(10, 10) = (10, 9) \oplus_{\text{WL}} (10, 1)$. Since $(10, 10)$ is preferred to $(5, 1)$, v replaces the latter width-length by the former and advertises it to u. At this moment in time there are two advertisements in transit, both from v to u: $(5, 1)$ and $(10, 10)$, in this order. When $(5, 1)$ arrives at u, this node elects $(5, 2) = (5, 1) \oplus_{\text{WL}} (5, 1)$, to the detriment of $(5, 7)$ previously learned from w, and advertises this width-length around the self-loop.

We now make the assumption that the $(10, 10)$ advertisement from v arrives at u before the $(5, 2)$ advertisement that is propagating around the self-loop. Upon reception of $(10, 10)$ from v, u extends this advertisement to candidate width-length $(5, 11) = (5, 1) \oplus_{\text{WL}} (10, 10)$, which is less preferred than candidate width-length $(5, 7)$ that u learns from w. Thus, $(5, 7)$ is re-elected and advertised around the self-loop. When the $(5, 2)$ advertisement in transit around the self-loop arrives back at u, this node extends it to $(5, 4) = (5, 2) \oplus_{\text{WL}} (5, 2)$, elects this width-length, and advertises it around the self-loop. At this instant in time, a forwarding loop is sustained around the self-loop, since u prefers the width-length learned from around the self-loop to the alternatives learned from v and w. The advertisement $(5, 4)$ travels around the self-loop and arrives back at u, giving rise to the election of $(5, 6) = (5, 2) \oplus_{\text{WL}} (5, 4)$, still preferred to width-length $(5, 7)$ that u learns from w. One last time, advertisement $(5, 6)$ travels around the self-loop, arrives back at u to become candidate width-length $(5, 8)$, which is less preferred than width-length $(5, 7)$ that u learned from w.

The protocol terminates in exactly the same state as in the first case with u electing width-length $(5, 7)$. Termination took longer because v leaks width-length $(5, 1)$ to u, which is desirable from the point of view of u, but does not belong to the stable state. This width-length is purged from the network by a slow process akin to count-to-infinity.

The problem solved by non-restarting vectoring protocols. If non-restarting vectoring protocols do not solve the problem of routing along optimal paths, then what problem do they solve? In the appendix, we show with generality that non-restarting vectoring protocols solve the problem of routing along optimal paths constrained by forwarding decisions at every node that depend exclusively on the destination of data-packets, as opposed to, for instance, on both the origin and destination of data-packets, or on the type of data to be conveyed. It is worthwhile to stress that this problem is well-defined independently of routing protocols. A result from [12] implies that it can also be solved with a generalization of Dijkstra's algorithm.

Back to the network of Figure 1, the solution to the problem of routing along optimal paths constrained by destination-based forwarding can be argued as follows. Data-packets with origin at w and destination at x are sent directly to x, since there is no other option. Data-packets with origin at v and destination at x can be forwarded to w or sent directly to x. If they are forwarded to w, then they travel along path vwx, width-length $(10, 10)$; if they are sent to x, then they travel along link vx, width-length $(5, 1)$. Since $(10, 10)$ is preferred to $(5, 1)$, v forwards data-packets to w. Data-packets with origin at u can be forwarded to either w or v. If they are forwarded to w, then they travel along path uwx, width-length $(5, 7)$; if they are forwarded to v, then, because v forwards data-packets to w regardless of whether they were originated at v or arrived there from u, they travel along path $uvwx$, width-length $(5, 11)$. Since $(5, 7)$ is preferred to $(5, 11)$, w forwards data-packets to w. This is exactly the stable state of a non-restarting vectoring protocol.

IV. Restarting vectoring protocols

Restarting vectoring protocols have been less scrutinized than non-restarting vectoring protocols in the networking literature. Therefore, we first illustrate their operation in a shortest-path context. We use the network of Figure 1, but assume that the protocol operates only on lengths. In the stable state, v and w both elect length 1 corresponding to their respective links to x, and u elects length 2, corresponding to path uvx.

Suppose that link uv fails. Then, u could elect length 4 learned from around the self-loop or length 7 learned from w, both less preferred than the previously elected length 2. Length 4 implies a forwarding loop around the self-loop and delayed termination, whereas length 7 is exempt from these undesirable conditions. However, u has no way of knowing the difference hiding behind these lengths. Restarting vectoring protocols simply rule that elected lengths cannot increase during a computation instance. Thus, when link uv fails, u is black-holed for destination x. Node u propagates this information around the self-loop and either waits for the destination to initiate a fresh computation instance, as in DSDV, or requests the destination to start such a computation at the earliest opportunity, as in Babel. Advertisements from the fresh computation instance supplant lengths elected during previous instances. Destination x advertises 0 to v and w, v elects 1, w elects 1, and advertises this length to v and u. The advertisement by w has no effect at v, but when it arrives at u, this node elects 7, which is its first elected length on this computation instance. The stable state has been reached.

We turn to the network of Figure 1 with the assumption that the protocol operates on width-lengths according to the shortest-widest path algebra. Destination x initiates a computation instance which leads to the same intermediate routing state as that of non-restarting vectoring protocols, marked in the figure. Node w elects $(10, 1)$, u elects $(5, 7)$, and there are two advertisements in transit: $(+\infty, 0)$ from x to v and $(10, 1)$ from w to v. We again distinguish two cases.

Case 1: advertisement $(10, 1)$, sent by w, arrives first at v. The execution of the protocol mimics that of a non-restarting vectoring protocol. Node u elects width-length $(5, 7)$ corresponding to path uwx.

Case 2: advertisement $(+\infty, 0)$, sent by x, arrives first at v. Then, v elects $(5, 1)$ and advertises this width-length to u.

Next, it elects $(10, 10)$ and advertises as well this width-length to u. When advertisement $(5, 1)$ arrives at u, this node elects $(5, 2)$, and advertises this width-length around the self-loop.

Presently, advertisement $(10, 10)$ is received at u. This node extends it to candidate width-length $(5, 11)$. The alternative candidate width-lengths available to u are $(5, 7)$, learned from w, and $(5, 4)$, learned from around the self-loop, if the advertisement sent by u has already traveled that length. All candidate width-lengths are less preferred than width-length $(5, 2)$, which was elected just before advertisement $(10, 10)$ is received at u. Therefore, u does not elect any of them; rather, it becomes a black-hole for x. The seriousness of this action is that u may not escape its black-hole condition during succeeding computation instances initiated by x, because advertisements may be received in the same order as before. This example suffices to show that, contrary to non-restarting vectoring protocols, in the absence of left-isotonicity, restarting vectoring protocols cannot be counted upon to deliver data-packets.

Why restarting vectoring protocols are inadequate in the absence of left-isotonicity. Restarting vectoring protocols presuppose that a routing solution can be found during a computation instance by repeatedly replacing elected attributes by more preferred ones. However, without left-isotonicity, an update of an elected attribute for a more preferred one at a node may imply an update of an elected attribute for a less preferred one at an in-neighbor during a single computation instance, a condition that is outlawed by the protocol, leaving the in-neighbor black-holed.

V. Implications

Many useful metrics do not satisfy left-isotonicity [1], [2], [13]. Whether or not they do, non-restarting vectoring protocols route correctly. The popularity of EIGRP bears witness to the suitability of these protocols. Its metric is a linear combination of inverse capacity and length, leading to a routing algebra that is not left-isotone and to paths that are not optimal, in general. Moreover, routing along better paths than those provided by non-restarting vectoring protocols requires a more sophisticated forwarding strategy than destination-based.

Restarting vectoring protocols avoid count-to-infinity and long-lived forwarding loops with low computational and state complexity. Initially proposed for wireless networks with highly dynamic topologies, these protocols have recently been considered in investigations that place the routing function directly in hardware [14], [15]. However, these protocols only work correctly on metrics that satisfy left-isotonicity.

Appendix

We show that the problem of routing along optimal paths subject to destination-based forwarding decisions, on the one hand, and the stable state of a non-restarting vectoring protocol, on the other, are both characterized by the same fixed-point algebraic equations. We consider the problem first. Let $E[u, t]$ be the attribute of a path followed by data-packets with origin at node u and destination at node t of a network. Clearly, $E[t, t] = \epsilon$, where ϵ is the identity of $\oplus$. Let v be an out-neighbor of u and denote by $a[uv]$ the attribute of link uv. If u would forward to v data-packets with destination in t, then the attribute of a path followed by data-packets with origin at u and destination at t would be $a[uv] \oplus E[v, t]$, and that is because *v forwards data-packets to t the same regardless of whether they have origin at v or arrive at v from u*. Therefore, the best choice available for a path from u to t is expressed by the equation

$$E[u, t] = \sqcap\{a[uv] \oplus E[v, t] |\ v \text{ an out-neighbor of } u\}, \quad (1)$$

where $\sqcap\{a_1, a_2, \ldots, a_n\}$ is a shorthand for $a_1 \sqcap a_2 \sqcap \cdots \sqcap a_n$.

Now, consider the stable state of a non-restarting vectoring protocol. Re-interpret $E[u, t]$ as the attribute elected at u to reach t in such a state. Clearly, $E[t, t] = \epsilon$. Let v be an out-neighbor of u. Since the state is stable, there are no advertisements in transit from v to u. The last advertisement that u received from v was $E[v, t]$. Then, u extended the advertisement into candidate attribute $a[uv] \oplus E[v, t]$ and elected an attribute from among the candidates. After u received the last advertisements from all its out-neighbors, the election at u is expressed by Equation (1).

References

[1] 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.

[2] Q. Dong, S. Banerjee, M. Adler, and A. Misra, "Minimum energy reliable paths using unreliable wireless links," in *Proc. 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, May 2005, pp. 449–459.

[3] V. C. M. Borges, M. Curado, and E. Monteiro, "Cross-layer routing metrics for mesh networks: Current status and research directions," *Computer Communications*, vol. 34, no. 6, pp. 681–703, 2011.

[4] J. L. Sobrinho, "An algebraic theory of dynamic network routing," *IEEE/ACM Transactions on Networking*, vol. 13, no. 5, pp. 1160–1173, October 2005.

[5] D. P. Bertsekas and R. Gallager, *Data Networks*, 2nd ed. New Jersey: Prentice-Hall, 1991, ISBN 0-13-200916-1.

[6] G. Malkin, *RIP Version 2*, IETF, November 1998, RFC 2453.

[7] D. Savage, J. Ng, S. Moore, D. Slice, P. Paluch, and R. White, *Cisco's Enhanced Interior Gateway Routing Protocol (EIGRP)*, May 2016, RFC 7868.

[8] Y. Rekhter, T. Li, and S. Hares, *A Border Gateway Protocol 4 (BGP-4)*, IETF, January 2006, RFC 4271.

[9] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," in *Proc. ACM SIGCOMM*, August 1994, pp. 234–244.

[10] J. Chroboczek, *The Babel Routing Protocol*, April 2011, RFC 6126.

[11] J. J. Garcia-Luna-Aceves, "Loop-free routing using diffusing computations," *IEEE/ACM Transactions on Networking*, vol. 1, no. 1, pp. 130–141, February 1993.

[12] J. L. Sobrinho and T. Griffin, "Routing in equilibrium," in *Proc. of the 19th International Symposium on the Mathematical Theory of Networks and Systems*, July 2010.

[13] Y. Yang and J. Wang, "Design guidelines for routing metrics in multihop wireless networks," in *Proc. IEEE INFOCOM*, April 2008, pp. 2288–2296.

[14] E. C. Molero, S. Vissicchio, and L. Vanbever, "Hardware-accelerated network control planes," in *Proc. 17th ACM Workshop on Hot Topics in Networks*, 2018, pp. 120–126.

[15] K.-F. Hsu, R. Beckett, A. Chen, J. Rexford, P. Tammana, and D. Walker, *Contra: A Programmable System for Performance-Aware Routing*, 2019, arXiv:1902.00849.

Acknowledgments

The author thanks José Brázio, Miguel Ferreira, and Nuno Lopes for their comments on earlier drafts of this letter.