Network Working Group
Internet-Draft
Intended status: Experimental
Expires: 3 April 2026

S. Mueller

30 September 2025

Direct Knowledge Extension to Distance Vector Routing
draft-mueller-dkextension-01

Abstract

Naive Distance Vector-based routing protocols like the Routing Information Protocol [RFC_1058] suffer from a phenomena called the "count to infinity problem" in the event of a network topology change. This Internet Draft extends a naive Distance Vector routing implementation with a simple flag that allows the network to recover quickly and reliably, with no chance of routing loops to occur.

Discussion Venues

This note is to be removed before publishing as an RFC.

Source for this draft and an issue tracker can be found at https://github.com/Sebastianmueller22/network-protocol.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 3 April 2026.

Copyright Notice

Table of Contents

1. Introduction

The count to infinity problem arises in distance vector routing protocols when a routing loop forms after a network topology change. In such scenarios, nodes within the loop continue to advertise routes to a failed node through each other. Misled by these advertisements, the nodes fail to recognize the network failure and continue routing traffic within the loop, incrementing the routing metrics until they reach an "infinity" value. At this point, the nodes assume a failure and cease routing traffic to the failed node. The infinity value imposes a limitation on the maximum network size, as the actual routing costs between distant nodes must remain below this threshold.

This document introduces a simple flag to distance vector routing protocols, addressing the count to infinity problem and eliminating the need for strict network size limits imposed by, for example, the Routing Information Protocol [RFC_1058]. Consequently, mechanisms such as split horizon with poisoned reverse and feasibility conditions become redundant. The proposed extension is designed to be compatible with "naive" Bellman-Ford based routing protocols like RIP2 [RFC_2453] rather than more sophisticated protocols like Babel [RFC_6126] , which were developed to tackle the count to infinity problem. Due to its simplicity, this extension should still be compatible with various distance vector-based routing protocols.

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Count to infinity problem - CTIP

Distance Vector Routing - DVR

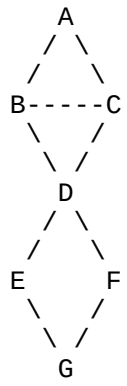
Routing Information Protocol - RIP

Direct Knowledge Bit - DKB

3. Distance Vector Routing

The basic principle of Distance Vector based routing is that nodes only exchange routing information with their direct neighbors. The local link costs are added to the advertised routing cost of a given, more distant node and the path with the lowest cost is chosen. Only the next hop to any given node or network needs to be saved by network participants.

Take this network as an example:



How do the network participants decide on the best path to G?

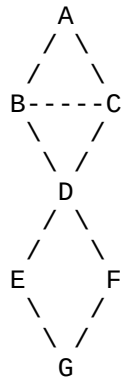
If the link cost between E and G is 10, while all other link costs are 1, E advertises a cost of 10 to D, while F advertises a cost of 1. D calculates the total cost to reach G via E as 11 by adding 1 to E’s advertised cost. However, since the path to G via F has a lower total cost of 2, D selects F as the next hop to G and updates its

routing table accordingly. In the next update cycle, D advertises its path to G through itself, with a total cost of 2. B and C add their local link cost of 1 and update their total cost to reach G to 3. In the next update cycle A selects either B or C as its next hop with a cost of 4.

We only considered the cost to reach G in this example (and will continue to do so for the remainder of this draft). This is simultaneously done for all other network participants though. Hopefully this quick example is enough to understand the principle of the routing algorithm. For interested readers, the according sections in the RFCs [RFC_6126] and [RFC_2453] are reccommended.

4. Count to infinity

Counting to an arbitrary infinity value is an attempt of naive DVR algorithms such as RIP to resolve routing loops after a network topology change.



The link costs are the same as in the previous example. We assume one basic protection against routing loops, namely split horizon with poisoned reverse. This means that, for example, when node A has node C as next hop to G in its routing table, it will not advertise this route to C (because the route goes via C, C knows about it). Routing loops can still emerge though, as we will see in this example.

When the link between F and G fails, a naive implementation of DVR sets the cost between F and G to an "infinity value" — a positive integer larger than the highest legal routing cost. According to the original RIP [RFC_1058], this value is set to 16. Consequently, F advertises the value of 16 to D. Due to the implementation of split horizon, D does not advertise the cost of 2 back to F. Instead, D selects E as the next best hop to G, updating its cost to reach G to 11.

B and C do not advertise their cost of 3 to D because of split horizon. In the following update cycle, B and C receive D’s updated cost of 11. In the same update cycle, they each advertise their previous cost of 3 to each other and to A, because those are not their next hop, which is still D. This causes B and C to believe they can route traffic to G via one another, without realizing their paths go through D. As a result, both add the local link cost of 1 to the routing cost of 3 and propagate this "new" route to A and D in the next cycle.

In subsequent update cycles, the cost of 1 is repeatedly added to the cost in the routing loop. It takes a significant number of cycles for this cumulative cost to eventually exceed 12 — the only remaining connection to G, at which point the loop resolves and the network converges to this path.

This takes a long time and in the case of a node failure, there is no way out of it other than to count up to an arbitrary infinity value (16 in this example). This value serves as an arbitrary threshold to terminate the process once it becomes excessively large. While effective in halting the loop, reaching this limit is time-consuming and restricts the range of permissible routing costs and thereby the possible size of the network.

5. The extension

This draft proposes the convention that if a node doesn’t receive an update message from one of its immediate neighbors for a certain amount of time, it tries to contact it with several messages which need to be acknowledged. If these remain unanswered, the node assumes the neighbor to be down. It then sends a triggered update to all other neighbors with the route to the down neighbor being set to -1 or another similarly impossible value. This can also be implemented as a separate flag in the routing information datagram. Although this is an adapted version of the "infinity value" of the RIP, it cannot be a positive number since we make no limitation on the network size. This infinity value signals the failure event to the rest of the network. All receiving nodes that aren’t direct neighbors to the failed node MUST perform the following steps:

* They write the infinity value into their routing table

* They stop routing traffic with the failed node as a target and report it to the sender as unreachable

* They ignore any regular updates that advertise a live route to the failed node

* They likewise send a triggered update with this infinity value to all their neighbors

This "bad news" travels with the full speed of triggered updates through the network since all regular advertisements reporting it to be up are ignored.

This goes on, until a node is reached that has the node in question as part of its direct neighbors. The receiving node then tries to reach the failed node. If it answers, the failure was actually a link failure, not a node failure, or the node recovered in the meantime. The node that receives a reply from its neighbor then immedately sends a triggered update with a "direct-knowledge-bit" (DKB) set. This is best implemented as a bit in the update datagram. Receiving nodes of such an update message MUST:

* Write this message into their routing table if they have an infinity value in there or an update message with higher cost and DKB set, or a regular routing cost (they didn’t know about the failure yet)

* Trigger an update message advertising this new route with the DKB set to all their neighbors

* Start a timeout and retain the DKB until the timeout expires. Any received infinity values within this time and for this node will be ignored

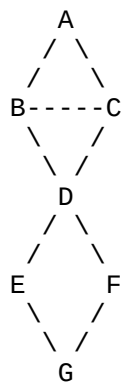
* Route traffic via the new path

So the "good news" travels with the full speed of triggered updates as well.

Since in the case of a failure no node will believe an update without direct knowledge of the nodes’ continuing or regained liveness, count to infinity cannot occur. All old routing information that potentially is no longer feasible is then discarded. The best path to the node is discovered as soon as the remaining neighbors hear about the failure through the triggered updates and the best remaining path is propagated with the speed of triggered updates as well.

6. Example

We again take as an example the following network topology:



In this example, contrary to before, we assume no other protection against routing loops than the proposed extension. So split horizon is not implemented, just to show that the extension is sufficient to avoid routing loops.

When the link between F and G fails, F sends an infinity value to D. D ignores updates reporting G as up from C,B and E and propagates the infinity value to B,C and E. E knows that G is its direct neighbor and upon receiving the infinity value, it checks if G is reachable. Since it is, it advertises a route to G via itself with the DKB set. Meanwhile B and C have sent the infinity value to A. D receives the advertisement with the DKB set, ignores the infinity values from F, B and C, and advertises this new route to them. B and C ignore the infinity value from A and advertise the new routing information with the DKB to A.

Now the network has converged to the remaining path to G.

7. Security Considerations

This document only tries to solve the count to infinity problem.

The inherent security risks of DVR, such as rogue nodes advertising routes that don’t exist, or routes for other nodes to themselves etc., apply here as well.

The additions proposed in this draft pose additional security risks. A rogue node could advertise all other nodes as being down whether they are its neighbor or not. This would effectively halt communication in the network for a short time and put significant strain on the network while all nodes report their neighbors to still be reachable.

This is not easily preventable, but can be mitigated with another convention where updates originating from a node need to cryptographically signed before sending. The public key infrastructure necessary for that would need to be implemented on a higher network level.

That way, repeated infinity values from the same node can be ignored for a certain time (which might be advisable anyway, in the context of frequently failing links).

8. IANA Considerations

This document has no IANA actions.

9. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC_1058] C., H., "Routing Information Protocol", DOI 10.17487/RFC1058 , June 1988, <https://www.rfc-editor.org/info/rfc1058>.

[RFC_2453] Malkin, G. S., "RIP Version 2", DOI 10.17487/RFC2453 , November 1998, <https://datatracker.ietf.org/doc/rfc2453/>.

[RFC_6126] Chroboczek, J., "The Babel Routing Protocol", DOI 10.17487/RFC6126 , April 2011, <https://www.rfc-editor.org/info/rfc6126>.

Author’s Address

Sebastian Mueller
Email: directknowledgeextension@gmx.de