Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: 18 September 2026

Q. Yang, Ed.
F. Zhu
V. Wen
Arista Networks Inc
H. Zheng
L. Yang
ByteDance Inc
17 March 2026

S-BFD Proxy
draft-yang-bfd-sbfd-proxy-03

Abstract

This document proposes an extension to Seamless Bidirectional Forwarding Detection (S-BFD).

The S-BFD initiator will send packets that carry extra information, and this enables reflector to act as a proxy, and respond with the extra information in consideration.

This document updates RFC 7880.

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 18 September 2026.

Copyright Notice

Table of Contents

1. Introduction 2
1.1. Requirements Language 3
2. S-BFD Proxy Reflector Overview 3
3. Packet format 4
4. S-BFD Proxy Reflector Procedures 6
5. S-BFD Proxy Initiator Procedures 7
6. SBFDInitiator State Machine 7
7. S-BFD Echo Function 7
8. Co-existence with Classical S-BFD Sessions 7
9. IANA Considerations 7
10. Security Considerations 7
11. References 8
11.1. Normative References 8
Acknowledgements 8
Authors' Addresses 8

1. Introduction

Seamless Bidirectional Forwarding Detection (S-BFD), as described in [RFC7880] and related documents, has defined a mechanism for liveliness detection of paths end-to-end.

However, in certain cases, such liveliness detection is not possible end-to-end. There could be a number of reasons. For example, the path termination point may not be able to perform S-BFD reflector functionality, or it is not possible for the path termination point to route back to the source IP address of the initiator.

In such cases, however, it may still be useful or essential for the S-BFD initiator to detect the liveliness of the path.

In this draft, we propose a mechanism for an intermediate node in the path to act as a proxy for the real end point, and the report S-BFD status would be for the entire path.

1.1. Requirements Language

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.

2. S-BFD Proxy Reflector Overview

Similar to regular S-BFD sessions, in an S-BFD proxy session, the initiator will determine, with help of configuration, which is outside the scope of this specification, a proxy reflector, along the path, to be the responder of the S-BFD packet. It then sends the S-BFD packet terminating at the proxy reflector, and the proxy reflector reflects the S-BFD packet with a status in consideration of the remaining path.

In order for the proxy reflector to reflect the status of the entire path, the initiator will have to include the information about the remaining path in the S-BFD packet payload (see section 4).

When the reflector receives the S-BFD packet, instead of just reflecting the packet as defined in RFC7880, it will retrieve the remaining path information from S-BFD packet, and inspect the health of the corresponding path (scope outside of this spec). It will then respond with the appropriate status back to the initiator.

It is possible that the initiator may have multiple paths passing through the proxy reflector. In order for the initiator to differentiate the different sessions, the proxy reflector will have to keep track of the discriminator combination. The reflector MAY remove the auxiliary TLV from the S-BFD payload, except when required otherwise. See section 4 for exception.

This is exemplified in Figure 1.

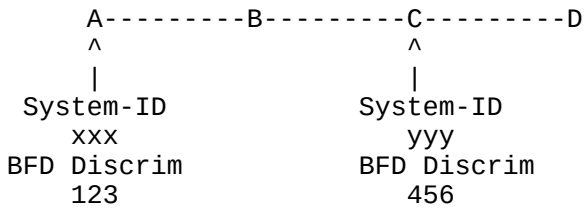

Figure 1: S-BFD Proxy

In this example, A is the head end, and has a path all the way to D. But for some reason, an end to end S-BFD session to D is not possible. Instead, A will attempt to have an S-BFD session using C as the proxy reflector.

A will encode the remaining path corresponding to the last segment, C-D, in the S-BFD packet payload. When C, acting as proxy reflector, receives the packet, it decodes the path information for C-D and inspects the health of that segment, through some means, and responds to A. A will interpret the response as the status of the entire tunnel end-to-end.

3. Packet format

S-BFD proxy reflector introduces new field in the BFD packet.

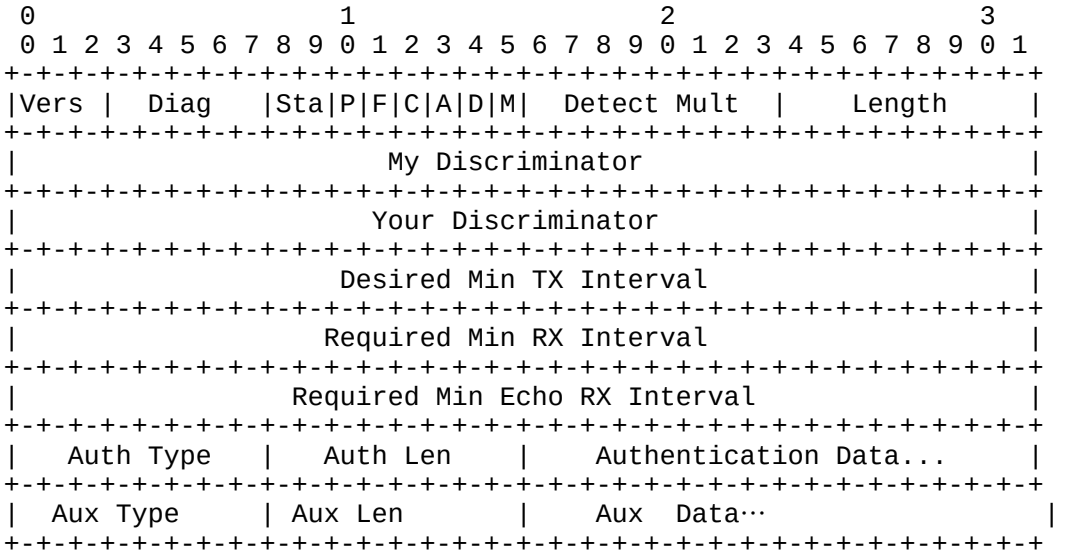

Figure 2: Packet format

The extra information is carried after the (optional) Auth Type, in the Aux TLVs.

Since there is no bit available in the 2nd 8-bit header to indicate the existence of this field, we will modify the decoding procedure as the following: if, after decoding to the end of S-BFD packet, including the optional Auth, the length consumed is less than the total Length field in the first word, the remaining bytes will be considered as Aux TLVs.

There might be multiple auxiliary TLVs from that point on. If at any point, the Aux Len indicates a length bigger than the remaining bytes, the packet will be considered malformed and MUST be discarded. Aux Len must have a minimum value of 2 (in bytes). Packet with Aux Len less than 2 MUST be discarded.

Otherwise, the Aux Len bytes will be consumed and its interpretation will depend on the Aux Type specification. If somehow the packet end is reached with still remaining length, the packet MUST be discarded.

The Aux Len denotes the length of the auxiliary path data. The Aux Lens from all auxiliary TLVs are added to the BFD length field.

The Aux Type takes 8 bits, with the highest bit as the ‘reflection bit’.

When the reflection bit is not set, the reflector which supports this specification is required to inspect the content, and reflects the packet accordingly. If the reflector does not recognize this particular type, it MUST drop it without reflecting back. This rule shall trump the next. I.e, if there are multiple Aux TLVs, and a mix of TLVs with reflection bit set and not set, the ‘reflection bit clear’ rule shall take precedence.

When this bit is set, i.e, for type in ranges of 128-255, it is required to reflect back the packet even without understanding the content of the type and TLV. Inspecting the TLV contents, when the reflector recognizes the type, is optional and reflecting status back is at the discretion of the reflector.

If authentication is included, then it is still assumed that the authentication is done on the entire packet, including the auxiliary TLV portion.

For each Aux Type, there SHOULD be at most one TLV associated with. If more than one is sent, undefined behavior may follow.

The following Aux types are defined:

Aux Type (8 bit):

0 reserved

1 MPLS label stack for purpose of additional check

2 MPLS label stack for purpose of return path

When the type is set to 1, it denotes a stack of labels. The labels may or may not have global significance, and its allocation and communication between the S-BFD initiator and S-BFD proxy reflector is outside the scope of this document. Each label will take the entire 32 bit (including TOS, BOS, and TTL field, but these may not be used). The purpose of type 1 is for proxy functionality, that the reflector may check reachability of the label stack

When the type is set to 2, it again denotes a stack of labels. The labels are encoded the same way as 1. Here the reflector is expected to use the label stack as the return path back to initiator, as versus going over the IP forwarding.

4. S-BFD Proxy Reflector Procedures

The reflector, or responder, upon reception of proxy S-BFD Control packets (type 1 above), after verifying the validity of the packets, will inspect payload and check the health of the remaining path, and respond with the appropriate status.

If the remaining tunnel path is down, the responder must set the bfd.SessionState to Down. Else it should set to Up. Notice here, when the remaining path is down, the responder should not drop the packet and let the initiator time out. This will help the initiator to detect the failure faster.

For backward compatibility purposes, we propose to set the bfd.SessionState to Down, as versus AdminDown, as specified in RFC7880.

In addition it shall also respond to the initiator with a diagnostic code of ‘6 -- Concatenated Path Down ‘ in the S-BFD reply.

The reflector, or responder, upon reception of S-BFD control packets with type 2 above, after verifying the validity of the packets, shall respond with the appropriate status. In addition, it shall encapsulate the packet with the specified label stack, and forward accordingly. The label stack here will represent a potentially traffic engineered path back to the initiator.

5. S-BFD Proxy Initiator Procedures

Changes to RFC7880 is when an initiator receives a packet with bfd.SessionState to be Down, it should immediately turn the S-BFD state to be Down.

6. SBFDInitiator State Machine

The following diagram provides the RECOMMENDED state machine for stateful SBFDInitiators.

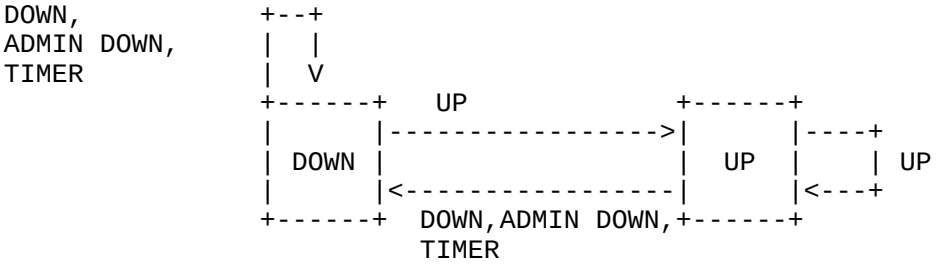

Figure 3: SBFDInitiator Finite State Machine

Note that the above state machine is different from the S-BFD specification [RFC7880]. This is because the proxy reflector may send a packet with DOWN state.

7. S-BFD Echo Function

Since the operation outlined here requires cooperation of the reflector, echo mode is not supported for S-BFD proxy.

8. Co-existence with Classical S-BFD Sessions

It is possible for S-BFD proxy sessions to coexist with regular S-BFD sessions, even between the same initiator-reflector pair.

9. IANA Considerations

This memo includes no request to IANA.

10. Security Considerations

The same security considerations as those described in RFC7880 will apply to this document.

11. References

11.1. 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/info/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/info/rfc8174>.

[RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <https://www.rfc-editor.org/info/rfc5880>.

[RFC7880] Pignataro, C., Ward, D., Akiya, N., Bhatia, M., and S. Pallagatti, "Seamless Bidirectional Forwarding Detection (S-BFD)", RFC 7880, DOI 10.17487/RFC7880, July 2016, <https://www.rfc-editor.org/info/rfc7880>.

[RFC7881] Pignataro, C., Ward, D., and N. Akiya, "Seamless Bidirectional Forwarding Detection (S-BFD) for IPv4, IPv6, and MPLS", RFC 7881, DOI 10.17487/RFC7881, July 2016, <https://www.rfc-editor.org/info/rfc7881>.

[RFC7882] Aldrin, S., Pignataro, C., Mirsky, G., and N. Kumar, "Seamless Bidirectional Forwarding Detection (S-BFD) Use Cases", RFC 7882, DOI 10.17487/RFC7882, July 2016, <https://www.rfc-editor.org/info/rfc7882>.

[RFC7885] Govindan, V. and C. Pignataro, "Seamless Bidirectional Forwarding Detection (S-BFD) for Virtual Circuit Connectivity Verification (VCCV)", RFC 7885, DOI 10.17487/RFC7885, July 2016, <https://www.rfc-editor.org/info/rfc7885>.

Acknowledgements

The authors of this document would like to thank Joseph Swaminathan, Zhen Xue, Hanchen Zheng, Kelvin Liu and Mamen Xu for the discussion and comments of this document.

Authors' Addresses

Qing Yang (editor)
Arista Networks Inc

Email: qyang@arista.com

Feng Zhu
Arista Networks Inc
Email: fzhu@arista.com

Victor Wen
Arista Networks Inc
Email: vwen@arista.com

Hanchen Zheng
ByteDance Inc
Email: hanchen.zheng@bytedance.com

Liu Yang
ByteDance Inc
Email: yangliu.jason@bytedance.com