ippm
Internet-Draft
Intended status: Experimental
Expires: November 19, 2017

H. Song, Ed.
T. Zhou
Huawei

May 18, 2017

On Scalability of In-situ OAM
draft-song-ippm-ioam-scalability-00

Abstract

This document describes several scalability issues in current in-situ OAM documents and proposes corresponding solutions. Specifically, we extend in-situ OAM to support more standard tracing data than is currently defined and add new features to avoid limitations on MTU, bandwidth, forwarding path length, and node processing capability.

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 http://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 November 19, 2017.

Table of Contents

1. Introduction

In-situ OAM (iOAM) [I-D.brockners-inband-oam-requirements] records OAM information within user packets while the packets traverse a network. The data types and data formats for in-situ OAM data records have been defined in [I-D.brockners-inband-oam-data]. We identify several scalability issues of the current iOAM specification and propose solutions in this draft.

2. Current iOAM Limitations

2.1. Data Type Limitation

Currently 11 data types and associated formats (including wide format and short format of the same data) are defined in [I-D.brockners-inband-oam-data] . The presence of data is indicated by a 16-bit bitmap in the "OAM-Trace-Type" field.

In the current specification only five bits are left to identify new data types. Moreover, some data is forced to be bundled together as a single unit to save bitmap space and pack data to the ideal size (e.g., the hop limit and the node id are bundled, and the ingress interface id and the egress interface id are bundled), regardless of

the fact that an application may only ask for a part of the data. Last but not the least, each data is forced to be 4-byte aligned for easier access, resulting in waste of header space in many cases.

Since the data plane bandwidth, the data plane packet processing, and the management plane data handling are all precious yet scarce resource, the scheme should strive to be simple and precise. The application should be able to control the exact type and format of data it needs to collect and analyze. It is conceivable that more types of data may be introduced in the future. However, the current scheme cannot support it after all the bits in the bitmap are used up.

Currently, bit 7 is used to indicate the presence of variable length opaque state snapshot data. While this data field can be used to store arbitrary data, the data is difficult to be standardized and another schema is needed to decode the data, which may lead to low data plane performance.

2.2. Node Data Size Limitation

The total size of data is limited by the MTU. When the number of required data types is large and the forwarding path length is long, it is possible that there is not enough space in the iOAM header to save the data. The current proposal is to label the overflow status and stop adding new node data to the packet, leading to loss of information.

Even if the header has enough space to hold the iOAM data, the overhead may be too large and consume too much bandwidth. For example, if we assume moderate 16 bytes of data per node, a path with length of 10 will need 160 bytes to hold the data. This will inflate small 64-byte packets by more than three times. Therefore, we need to limit the iOAM data overhead without sacrificing the data collection capability.

2.3. Node Processing Limitation

iOAM can designate the flow to add the iOAM header and collect data on the flow forwarding path. The flow can have arbitrary granularity. However, processing the data can be a heavy burden for the network nodes, especially when some data needs to be calculated by the node (e.g., the transit delay). If the flow traffic is heavy, the node may not be able to handle the iOAM processing so many performance issues may occur, such as long latency and packet drop.

Although it is good for the OAM applications to gain the detailed information on every packet at every node, in many cases, such

information is often repetitive and redundant. The large quantity of data would also burden the management plane which needs to collect and stream the data for analytics. It is also possible that some nodes cannot provide the requested data at all. So a trade-off is needed to balance the performance impact and the data availability and completeness.

3. Scalable Data Type Extension

Based on the observation in Section 2.1, we propose a method for data type encoding which can solve the current limitation and address future data requirements.

3.1. Data Type Bitmap

Bitmap is simple and efficient data structure for high performance data plane implementation. The base bitmap size is kept to be 16 bits. We use one bit to indicate a single type of data in a single format. The last bit in the bitmap (i.e., bit 15), if set, is used to indicate the presence of the next data type bitmap, which is 32 bits long. In the second bitmap, bit 31 is again reserved to indicate a third bitmap, and so on. With each extra bitmap, 31 more data types can be defined.

Figure 1 shows an example of the in-situ OAM header format with two extended OAM trace type fields. Except the OAM Trace Type fields, all other fields remain the same as defined in [I-D.brockners-inband-oam-data].

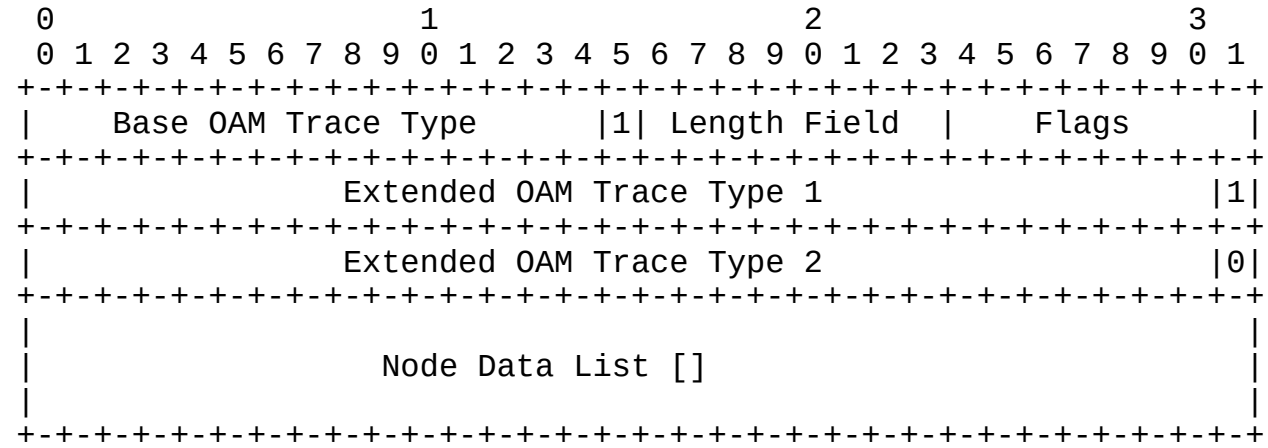

Figure 1: Extended OAM Trace Type Header Format

The specification of the Base OAM Trace Type is the same as the OAM Trace Type in [I-D.brockners-inband-oam-data] except the last bit, which is defined as follows:

- Bit 15: When set indicates presence of next bit map.

The OAM trace type fields are labeled as Base OAM Trace Type, Extended OAM Trace Type 1, Extended OAM Trace Type 2, and so on. The Base OAM Trace Type is always present. If no data type is asked by the application in Extended OAM Trace Type n and beyond, then the last bit in the previous bitmap is set to 1 and these extended fields are not included in the header. On the other hand, to eliminate ambiguity, if any data is asked for by the application in Extended OAM Trace Type n, then Extended OAM Trace Type 1 to (n-1) must be included in the header, even though no data type in these bitmaps are needed (i.e., all zero bitmap except the last bit).

The actual data in a node is packed together in the same order as listed in the OAM Trace Type bitmap. Each node is padded to be the multiple of 4 bytes.

3.2. Scalable Data Type Extension Use Cases

New types of data can be potentially added and standardized, which demand new bits allocated in the OAM Trace Type bitmaps. Some examples are listed here.

- Metered flow bandwidth.
- Time gap between two consecutive flow packets.
- Remaining time budget to the packet delivery deadline.
- Buffer occupancy on the Node.
- Queue depth on each level of hierarchical QoS queues.
- Packet jitter at the Node.
- Other node statistics.

3.3. Consideration for Data Packing

The length of each data must be the multiple of 2 bytes. However, allowing different data type to have different length, while efficient in storage, makes data alignment and packing difficult.

If we can define the maximum number of data types that can be carried per packet, the offset of each data in the node can be pre-calculated and carried in the iOAM header. The overhead can be justified by the overall space saving of the node data list. Otherwise, each data's offset in the node must be calculated in each device, with the help of a table which stores the size of each data type. We can also arrange the bitmap to reflect the data availability order in the system (e.g., the bit for egress_if_id must be after the bit for ingress_if_id), so in a pipeline-based system, the required data can be packed one after one.

4. Segment In-situ OAM

Based on the observation in Section 2.2, we propose a method to limit the size of the node data list.

4.1. Segment and Hops

A hop is a node on a flow's forwarding path which is capable of processing iOAM data. A segment is a fixed number hops on a flow's forwarding path. While working in the "per hop" mode, the segment size (SSize) and the remaining hops (RHop), is added to the iOAM header at the edge. Initially, RHop is equal to SSize. At each hop, if RH is not zero, the node data is added to the node data list at the corresponding location and then RH is decremented by 1. If RH is equal to 0 when receiving the packet, the node needs to remove (in incremental trace option) or clear (in pre-allocated trace option) the iOAM node data list and reset RHop to SSize. Then the node will add its data to the node data list as if it is the edge node.

Figure 2 shows the proposed in-situ OAM header format. The last bit (bit 31) in the Flags field is used to indicate the current header is a segment iOAM header. In this context, the third byte of the first word is partitioned into two 4-bit piece. The first piece is used to save the segment size and the second piece is used to save the remaining hops. This limits the maximum segment size to 15.

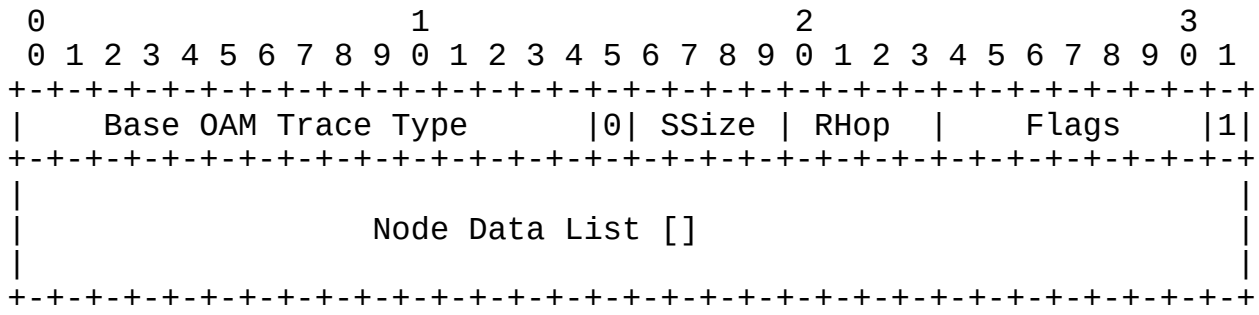

Figure 2: Segment iOAM Header Format

4.2. Considerations for Data Handling

At any hop when RHop is equal to 0, the node data list is copied from the iOAM header. The data can be encapsulated and reported to the controller or the edge node as configured. The encapsulation and report method is beyond the scope of this draft but should be comply with the method used by the iOAM edge node.

The actual size of the last segment may not be equal to SSize but this is not a problem.

4.3. Segment iOAM Use Cases

Segment iOAM is necessary in the following example scenarios:

- Segment iOAM can be used to detect at which segment the flow packet is dropped. If the SSize is set to 1, then the exact drop node can be identified. The iOAM data before the dropping point is also retained.

- The path MTU allows to add at most k node data in the list to avoid fragmentation. Therefore SSize is set to k and at each hop where RHop is 0, the node data list is retrieved and sent in a standalone packet.

- A flow contains mainly short packets and travels a long path. It would be inefficient to keep a large node data list in the packet so the network bandwidth utilization rate is low. In this case, segment iOAM can be used to limit the ratio of the iOAM data to the flow packet payload.

- The network allows at most n bytes budget for the iOAM data. There is a tradeoff between the number of data types that can be collected and the number of hops for data collecting. The segment

size is therefore necessary to meet the application's data requirement (i.e., SSize * Node Data Size < n).

5. In-situ OAM Sampling and Data Validation

Based on the observation in Section 1.3, the source edge node should be able to define either the period or the probability to add the iOAM header to the selected flow packet. In this way, only a subset of the flow/sec packets would carry the OAM data, which not only reduces the overall iOAM data quantity but also reduces the processing work load of the network nodes.

5.1. Valid Node Bitmap and Valid Data Bitmap

It is possible that even an iOAM capable node will not add data to the node data list as requested. In some cases, a node can be too busy to handle the data request or some types of the requested data is not available. Therefore, we propose to add two bitmaps, a valid node bitmap and a valid data bit, to the iOAM specification.

The Node Valid Bitmap is inserted before the Node Data List as shown in Figure 3. Each bit in the bitmap corresponds to a hop on the packet's forwarding path. The bits are listed in the same order as the hop on the packet's forwarding path. The bitmap is cleared to all zero at first. If a hop can add data to the Node Data List, the corresponding bit in Node Valid Bitmap is set to 1. The bit location for a hop can be calculated from the length field (e.g, the bit index is equal to SSize-RHop).The valid node data items in the node data list is equal to the number of 1's in the Node Valid Bitmap.

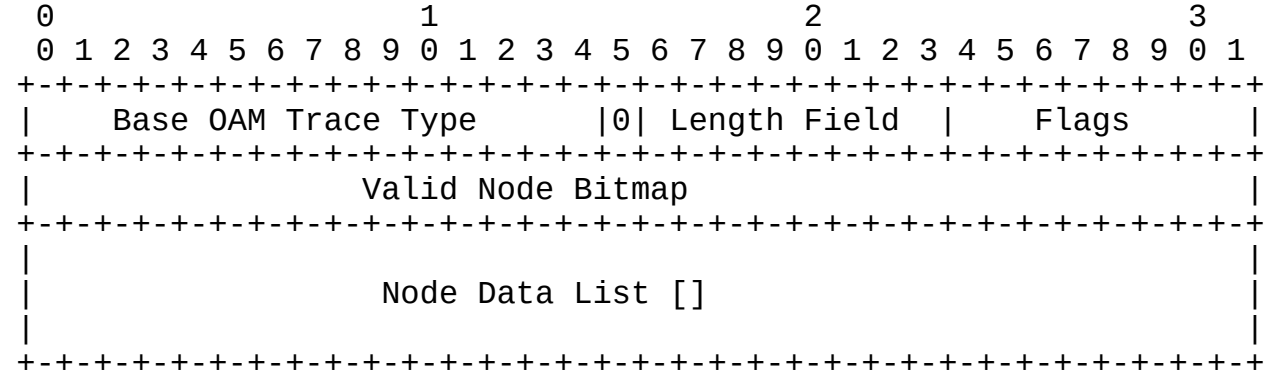

Figure 3: Segment iOAM Header Format

For each node data in the node data list, a Valid Data Bitmap is added before the node data. The number of bits in the Valid Data

Bitmap is equal to the number of 1's in the OAM Trace Type bitmaps (excluding the next trace type bitmap indicator bits). When the bit is set, the corresponding data is valid in the node; otherwise, the corresponding data is invalid so the management plane should ignore it after the data is collected.

The size of the bitmap can be padded to two or four bytes, which allow up to 16 or 32 types of data to be included in a node.

5.2. iOAM Sampling and Data Validation Use Cases

We give some examples to show the usefulness of in-situ OAM sampling and data validation features.

- An application needs to track a flow's forwarding path and knows the path will not change frequently, so it sets a low sampling rate to periodically insert the iOAM header to request the node ID.

- In a heterogeneous data plane, some nodes support to provide data x but the other nodes do not support it. However, an application is still interested in collecting data x if available. In this case, iOAM header can still be configured to ask for data x but the nodes that cannot provide the data simply invalidates it by resetting the corresponding bit in the valid data bitmap.

- Multiple sampling rate and multiple data request schema can be defined for a flow based on applications requirements and the data property, so for a flow packet, there can be no iOAM header or different iOAM headers. The node does not need to process all data all the time.

- For security reason, a node decides to not participate in the iOAM data collection. While it processes the other iOAM header fields as usual, it does not set the node valid bit in the Node Valid Bitmap and add node data to the Node Data List.

6. Security Considerations

There is no extra security considerations beyond those have been identified by in-situ OAM protocol.

7. IANA Considerations

This memo includes no request to IANA.

8. Acknowledgments

9. Contributors

The document is inspired by numerous discussions with James N. Guichard. He also provided significant comments and suggestions to help improve this document.

10. Informative References

[I-D.brockners-inband-oam-data]
Brockners, F., Bhandari, S., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mizrahi, T., Mozes, D., Lapukhov, P., and R. <>, "Data Formats for In-situ OAM", draft-brockners-inband-oam-data-02 (work in progress), October 2016.

[I-D.brockners-inband-oam-requirements]
Brockners, F., Bhandari, S., Dara, S., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mozes, D., Mizrahi, T., <>, P., and r. remy@barefootnetworks.com, "Requirements for In-situ OAM", draft-brockners-inband-oam-requirements-02 (work in progress), October 2016.

Authors' Addresses

Haoyu Song (editor)
Huawei
2330 Central Expressway
Santa Clara, 95050
USA

Email: haoyu.song@huawei.com

Tianran Zhou
Huawei
156 Beiqing Road
Beijing, 100095
P.R. China

Email: zhoutianran@huawei.com