NETCONF
Internet-Draft
Intended status: Standards Track
Expires: May 15, 2018
November 11, 2017

G. Zheng
T. Zhou
A. Clemm
Huawei

UDP based Publication Channel for Streaming Telemetry
draft-ietf-netconf-udp-pub-channel-01

Abstract

This document describes a UDP-based publication channel for streaming telemetry use to collect data from devices. A new shim header is proposed to facilitate the distributed data collection mechanism which directly pushes data from line cards to the collector. Because of the lightweight UDP encapsulation, higher frequency and better transit performance can be achieved.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Table of Contents

1. Introduction

Streaming telemetry refers to sending a continuous stream of operational data from a device to a remote receiver. This provides an ability to monitor a network from remote and to provide network analytics. Devices generate telemetry data and push that data to a collector for further analysis. By streaming the data, much better performance, finer-grained sampling, monitoring accuracy, and bandwidth utilization can be achieved than with polling-based alternatives.

Sub-Notif [I-D.ietf-netconf-subscribed-notifications] and YANG-Push [I-D.ietf-netconf-yang-push] defines a mechanism that allows a collector to subscribe to updates of YANG-defined data that is maintained in a YANG [RFC7950] datastore. The mechanism separates the management and control of subscriptions from the transport that

is used to actually stream and deliver the data. Two transports have been defined so far, NETCONF [RFC6241] and RESTCONF [RFC8040].

While powerful in its features and general in its architecture, in its current form the mechanism needs to be extended to stream telemetry data at high velocity from devices that feature a distributed architecture. The transports that have been defined so far, NETCONF and RESTCONF, are ultimately based on TCP (Transmission Control Protocol) and lack the efficiency needed to stream data continuously at high velocity. A lighter-weight, more efficient transport, e.g. a transport based on UDP (User Datagram Protocol) is needed.

- o Firstly, data collector will suffer a lot of TCP connections from, for example, many line cards equipped on different devices.

- o Secondly, as no connection state needs to be maintained, UDP encapsulation can be easily implemented by hardware which will further improve the performance.

- o Thirdly, because of the lightweight UDP encapsulation, higher frequency and better transit performance can be achieved, which is important for streaming telemetry.

This document specifies a higher-performance transport option for YANG-Push that leverages UDP. Specifically, it facilitates the distributed data collection mechanism described in [I-D.zhou-netconf-multi-stream-originators]. In the case of data originating from multiple line cards, the design requires data to be internally forwarded from those line cards to the push server, presumably on a main board, which then combines the individual data items into a single consolidated stream. The centralized data collection mechanism can result in a performance bottleneck, especially when large amounts of data are involved. What is needed instead is the support for a distributed mechanism that allows to directly push multiple individual substreams, e.g. one from each line card, without needing to first pass them through an additional processing stage for internal consolidation, but still allowing those substreams to be managed and controlled via a single subscription. The proposed UDP publication channel natively supports the distributed data collection mechanism.

While this document will focus on the data publication channel, the subscription can be used in conjunction with the mechanism proposed in [I-D.ietf-netconf-yang-push] with necessary extensions [I-D.zhou-netconf-multi-stream-originators].

2. Terminology

Streaming telemetry: refers to sending a continuous stream of operational data from a device to a remote receiver. This provides an ability to monitor a network from remote and to provide network analytics.

3. Solution Overview

The typical distributed data collection solution is shown in Fig. 1. The Subscriber cannot see the Agents directly, so it will send the Global Subscription information to the Master (e.g., main board). When receiving a Global Subscription, the Subscription Server decomposes the subscription request into multiple Component Subscriptions, each involving data from a separate internal telemetry source, for example a line card. The Component Subscriptions are distributed to the Component Subscription Server located in Agents. Subsequently, each Agent generates its own stream of telemetry data, collecting and encapsulating the packets per the Component Subscription and streaming them to the designated Collector.This distributed data collection mechanism may form multiple Publication Channels between the data originators and the Collector. The Collector is able to assemble many pieces of data associated with one Global Subscription.

The Publication Channel supports the reliable data streaming, for example for some alarm events. The Collector has the option of deducing the packet loss and the disorder based on the information carried by the notification data. And the Collector will decide the behavior to request retransmission. The Collector can send the retransmission request to the subscriber server for further processing.

The rest of the draft describes the UDP based publication channel.

```
                retransmission +    + Global
                request        |    | Subscription
                        +------------------------+
                        |      |    |  Master    |
                        |   +--v----v--------+   |
                        |   |  Subscription  |   |
                        |   |  Server        |   |
                        |   +--+----+-----+--+   |
                        |      |    |     |      |   internal
       Component        +------------------------+   subscription
       Subscription            |    |     |          distribution
               +---------------+    |     +--------------+
               |                    |                    |
     +------------------+  +------------------+  +------------------+
     |         |        |  |        |         |  |       |          |
     | +-------v------+ |  | +------v-------+ |  | +-----v--------+ |
     | | Component    | |  | | Component    | |  | | Component    | |
     | | Subscription | |  | | Subscription | |  | | Subscription | |
     | | Server       | |  | | Server       | |  | | Server       | |
     | +--------------+ |  | +--------------+ |  | +--------------+ |
     |       Agent 1    |  |      Agent 2     |  |        Agent n   |
     +---------+--------+  +--------+---------+  +----------+-------+
               |                    |                       |
               |                    | Publication Channel   |
               +--------------+     |     +-----------------+
                              |     |     |
                            +-v-----v-----v-+
                            |               |
                            |   Collector   |
                            |               |
                            +---------------+
```

Fig. 1 Distributed Data Collection

4. UDP Transport for Publication Channel

4.1. Design Overview

As specified in YANG-Push, the telemetry data is encapsulated in the NETCONF/RESTCONF notification message, which is then encapsulated and carried in the transport protocols, e.g. TLS, HTTP2. The following figure shows the overview of the UDP publication message structure.

- Next to the UDP encapsulation, the DTLS layer is to provide reusable security and authentication functions over UDP.

o The Message Header contains information that can facilitate the message transmission before de-serializing the notification message.

o Notification Message is the encoded content that the publication channel transports. The common encoding method includes GPB [1], CBOR [RFC7049], JSON, and XML. [I-D.ietf-netconf-notification-messages] describes the structure of the Notification Message for both single notification and multiple bundled notifications.

```
                      +--------------+
                      | Notification |
                      | Message      |
                      +--------------+

                      +--------------+
                      |   Message    |
                      |   Header     |
                      +--------------+

                      +--------------+
                      |     DTLS     |
                      +--------------+

                      +--------------+
                      |      UDP     |
                      +--------------+
```

Fig. 2 UDP Publication Message Overview

4.2. Data Format of the Message Header

The Message Header contains information that can facilitate the message transmission before de-serializing the notification message. The data format is shown as follows.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-------+---------------+-------+-------------------------------+
| Vers. |    Flag       |  ET   |      Length                   |
+-------+---------------+-------+-------------------------------+
|                      Notification-Time                        |
+---------------------------------------------------------------+
|                      Message-Generator-ID                     |
+---------------------------------------------------------------+
~                      Options                                  ~
+---------------------------------------------------------------+
```

Fig. 3 Message Header Format

The Message Header contains the following field:

- o Vers.: represents the PDU (Protocol Data Unit) encoding version. The initial version value is 0.

- o Flag: is a bitmap indicating what features this packet has and the corresponding options attached. Each bit associates to one feature and one option data. When the bit is set to 1, the associated feature is enabled and the option data is attached. The sequence of the presence of the options follows the bit order of the bitmap. In this document, the flag is specified as follows:

  * bit 0, the reliability flag;

  * other bits are reserved.

- o ET: is a 4 bits identifier to indicate the encoding type used for the Notification Message. 16 types of encoding can be expressed:

  * 0: GPB;

  * 1: CBOR;

  * 2: JSON;

  * 3: XML;

  * others are reserved.

- o Length: is the total length of the message, measured in octets, including message header.

- o Message-Generator-ID: is a 32-bit identifier of the process which created the message notification. This allows disambiguation of an information source, such as the identification of different line cards sending the notification messages.

- o Notification-Time: is the time at which the message leaves the exporter, expressed in seconds since the UNIX epoch of 1 January 1970 at 00:00 UTC, encoded as an unsigned 32-bit integer.

- o Options: is a variable-length field. The details of the Options will be described in the respective sections below.

4.3. Options

The order of packing the data fields in the Options field follows the bit order of the Flag field.

4.3.1. Reliability Option

The UDP based publication transport described in this document provides two streaming modes, the reliable mode an the unreliable mode, for different SLA (Service Level Agreement) and telemetry requirements.

In the unreliable streaming mode, the line card pushes the encapsulated data to the data collector without any sequence information. So the subscriber does not know whether the data is correctly received or not. Hence no retransmission happens.

The reliable streaming mode provides sequence information in the UDP packet, based on which the subscriber can deduce the packet loss and disorder. Then the subscriber can decide whether to request the retransmission of the lost packets.

In most case, the unreliable streaming mode is preferred. Because the reliable streaming mode will cost more network bandwidth and precious device resource. Different from the unreliable streaming mode, the line card cannot remove the sent reliable notifications immediately, but to keep them in the memory for a while. Reliable notifications may be pushed multiple times, which will increase the traffic. When choosing the reliable streaming mode or the unreliable streaming mode, the operate need to consider the reliable requirement together with the resource usage.

When the reliability flag bit is set to 1 in the Flag field, the following option data will be attached

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------------------------------------------------------+
|            Notification ID                                    |
+---------------------------------------------------------------+
|            Previous Notification ID                           |
+---------------------------------------------------------------+
```

Fig. 4 Reliability Option Format

The notification ID is generated continuously by the message generator. Different subscribers share the same notification ID sequence. Current ID and previous ID will be added in the packets.

For example, there are two subscriber A and B,

- o Notification IDs for the generator are : [1, 2, 3, 4, 5, 6, 7, 8, 9], in which Subscriber A subscribes [1,2,3,6,7] and Subscriber B subscribes [1,2,4,5,7,8,9].

- o Subscriber A will receive : [0,1][1,2][2,3][3,6][6,7].

- o Subscriber B will receive : [0,1][1,2][2,4][4,5][5,7][7,8].

4.4. Data Encoding

Subscribed data can be encoded in GPB, CBOR, XML or JSON format. It is conceivable that additional encodings may be supported as options in the future. This can be accomplished by augmenting the subscription data model with additional identity statements used to refer to requested encodings.

Implementation may support different encoding method per subscription. When bundled notifications is supported between the publisher and the receiver, only subscribed notifications with the same encoding can be bundled as one message.

5. Congestion Control

While efficient, UDP has no build-in congestion control mechanism. It is not recommended to use the UDP based publication channel over congestion-sensitive network paths. The deployments require the communications from exporters to collectors are always congestion controllable, i.e., the transport is over dedicated links or the streaming rate can be limited.

6. IANA Considerations

TBD

7. Security Considerations

TBD

8. Acknowledgements

The authors of this documents would like to thank Eric Voit, Tim Jenkins, and Huiyang Yang for the initial comments.

9. References

9.1. 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>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <https://www.rfc-editor.org/info/rfc7049>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

9.2. Informative References

[I-D.ietf-netconf-notification-messages] Voit, E., Bierman, A., Clemm, A., and T. Jenkins, "Notification Message Headers and Bundles", draft-ietf-netconf-notification-messages-02 (work in progress), October 2017.

[I-D.ietf-netconf-subscribed-notifications]
Voit, E., Clemm, A., Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Custom Subscription to Event Streams", draft-ietf-netconf-subscribed-notifications-07 (work in progress), October 2017.

[I-D.ietf-netconf-yang-push]
Clemm, A., Voit, E., Prieto, A., Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "YANG Datastore Subscription", draft-ietf-netconf-yang-push-11 (work in progress), October 2017.

[I-D.zhou-netconf-multi-stream-originators]
Zhou, T., Zheng, G., Voit, E., Clemm, A., and A. Bierman, "Subscription to Multiple Stream Originators", draft-zhou-netconf-multi-stream-originators-00 (work in progress), October 2017.

9.3. URIs

[1] https://developers.google.com/protocol-buffers/

Appendix A. Change Log

(To be removed by RFC editor prior to publication)

A.1. draft-ietf-zheng-udp-pub-channel-00 to v00

- Modified the telemetry header format.
- Add a section on the Authentication Option.
- Cleaned up the text and removed unnecessary TBDs.

A.2. v01

- Removed the detailed description on distributed data collection mechanism from this document. Mainly focused on the description of a UDP based publication channel for telemetry use.
- Modified the telemetry header format.

Authors' Addresses

Guangying Zheng
Huawei
101 Yu-Hua-Tai Software Road
Nanjing, Jiangsu
China

Email: zhengguangying@huawei.com

Tianran Zhou
Huawei
156 Beiqing Rd., Haidian District
Beijing
China

Email: zhoutianran@huawei.com

Alexander Clemm
Huawei
2330 Central Expressway
Santa Clara, California
USA

Email: alexander.clemm@huawei.com