NETCONF
Internet-Draft
Intended status: Standards Track
Expires: February 25, 2018

G. Zheng
T. Zhou
A. Clemm
Huawei

August 24, 2017

UDP based Publication Channel for Streaming Telemetry
draft-zheng-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].

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 February 25, 2018.

Table of Contents

1. Introduction 2
2. Terminology 4
3. Solution Overview 4
4. UDP Transport for Publication Channel 6
4.1. Data Format 6
4.2. Options 8
4.2.1. Reliability Option 8
4.2.2. Authentication Option 9
4.3. Data Encoding 10
5. IANA Considerations 10
6. Operational Considerations 10
7. Security Considerations 10
8. Acknowledgements 10
9. References 10
9.1. Normative References 10
9.2. Informative References 11
9.3. URIs 12
Appendix A. An Appendix 12
Authors' Addresses 12

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. Specifically, there are two aspects that need to be addressed:

1. 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.

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

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

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

2. The current design involves a single push server. 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. This centralized data collection mechanism can result in a performance bottleneck, especially when large amounts of data are involved. What is needed instead is 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.

This document specifies a distributed data collection mechanism which can directly push data from line cards to a collector by using a UDP based publication channel. Specifically, a higher-performance transport option for YANG-Push that leverages UDP is specified.

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.

Although the distributed data streaming from device line cards is one typical scenario that the proposed UDP based publication channel can be useful, the proposal is general enough to fit more scenarios that require UDP transport for data collections, e.g. the IoT (Internet of Things) use case.

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.

Component subscription: A subscription that defines the data from each individual entity which is managed and controlled by a single subscription server.

Subscription agent: An agent that streams telemetry data per the terms of a component subscription.

3. Solution Overview

The typical distributed data collection solution is shown in figure 1. The subscription server located in the main board receives the subscription requests or configurations. It may be colocated, not necessary, with a NETCONF server which interacts with outside clients. When receiving a subscription request, the subscription server decomposes the subscription into multiple component subscriptions, each involving data from a separate internal telemetry source, for example a line card. The component subscriptions are distributed within the device to the subscription agents located in line cards. Subsequently, each line card generates its own stream of telemetry data, collecting and encapsulating the packets per the component subscription and streaming it to the designated data collector.

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

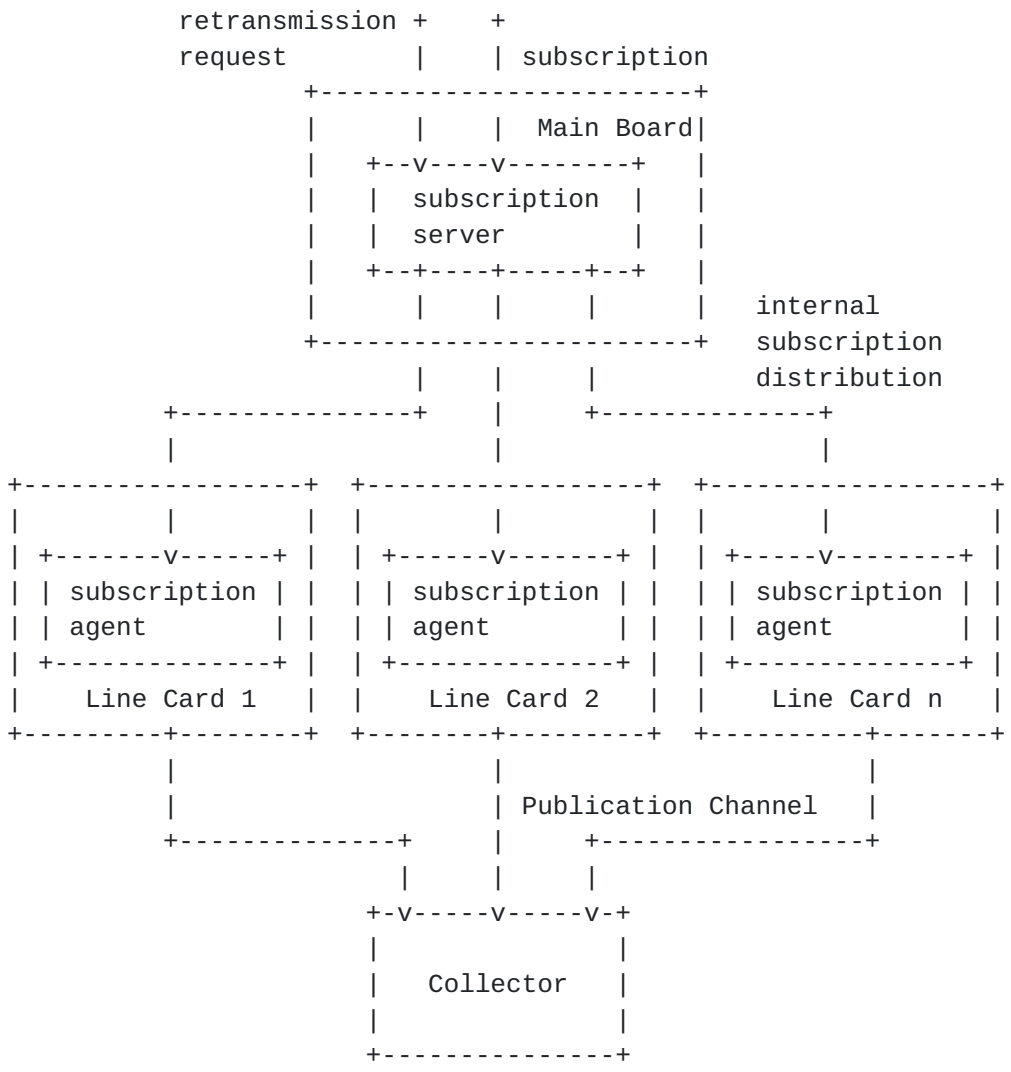
Subscription server and subscription agents interact with each other in several ways:

- Subscription agents need to have a registration or announcement handshake with the subscription server, so the subscription server is aware of them and of lifecycle events (such as subscription agents appearing and disappearing).

- The subscription server relays the component subscriptions to the subscription agents.

- The subscription agents indicate status of component subscriptions to the subscription server. The status of the overall "master" subscription is maintained by the subscription server. The subscription server is also responsible for notifying the subscriber in case of any problems of component subscriptions.

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

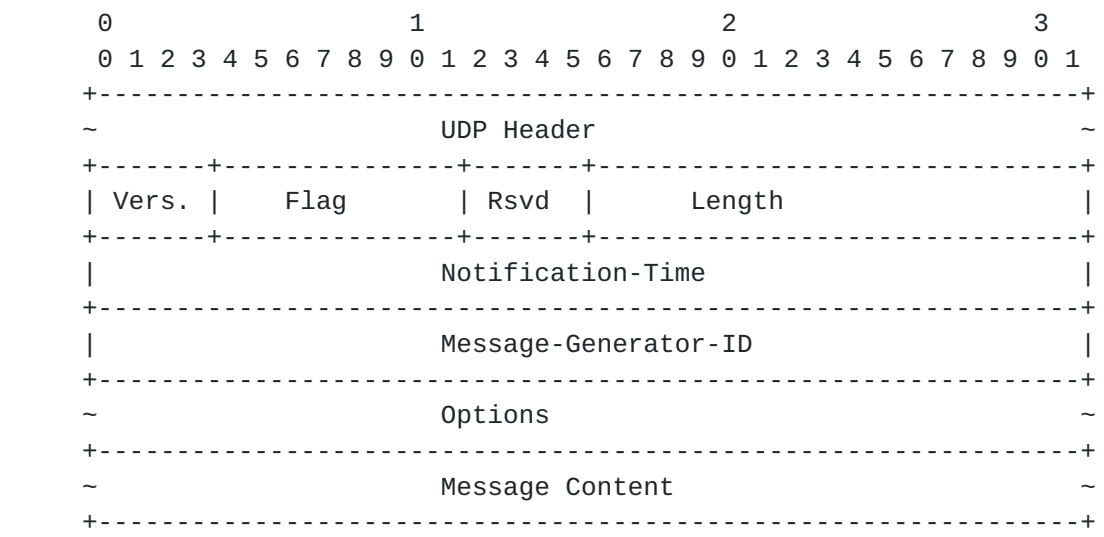


4. UDP Transport for Publication Channel

In [I-D.voit-netconf-notification-messages], the transport independent message header is proposed for the notification use. The following shim header refers to and implements that message header definition.

4.1. Data Format

The data format of the UDP based based publication transport is shown as follows.



Right after the UDP header, a simple inform header is attached to carry the necessary information with regard to the streaming mode.

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

- The 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, 2 flags are specified as follows:

  * bit 0, the reliability flag;

  * bit 1, the authentication flag;

  * other bits are reserved.

- The Length field is the total length of the message, measured in octets, including message header.

- The 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.

- The 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.

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

After the inform header is the real content which is encoded. The actual encoding is based on the subscription, e.g., in binary with GPB [1] or CBOR [RFC7049].

4.2. Options

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

4.2.1. Reliability Option

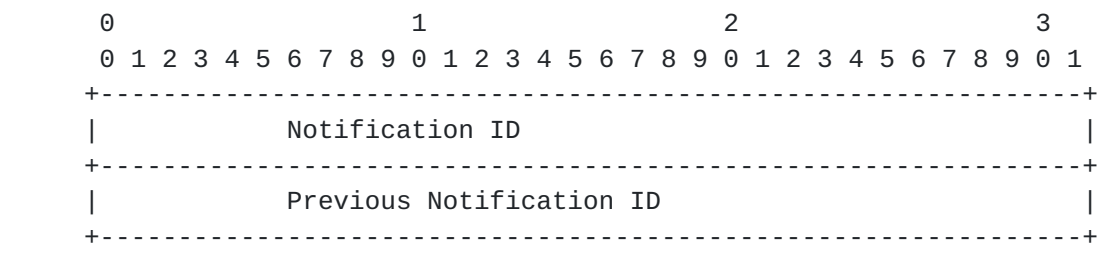
The UDP based publication transport described in this document provides two streaming modes, the reliable mode an the unreliable mode, for different SLA (Service Level Agreement) and telemetry requirements.

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



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,

- 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].

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

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

4.2.2. Authentication Option

When the authentication flag bit is set to 1 in the Flag field, a 24 octets data field will be included in the Options. The message is signed, and the signature is filled in the 24 octets Authentication Option field. So that a receiver can verify the authenticity of the message.

HMAC [RFC2104] defines a mechanism for message authentication using cryptographic hash functions. Any message digest algorithm can be used, but the cryptographic strength of HMAC depends on the properties of the underlying hash function. As suggested by [RFC6151], new protocol designs should not employ HMAC-MD5 [RFC2202]. Alternatives to HMAC-MD5 include HMAC-SHA256 [RFC4231] and AES-CMAC [RFC4493].

Implementations permit multiple acceptable algorithms, while the HMAC-SHA256 algorithm is mandatory in this document. The resulting message digest (output of HMAC) is truncated to 24 octets, which is the 192 leftmost bits of the HMAC computation, to fit the size of the Authentication Option field. It is recommended in [RFC2104] that the truncated output length should be not less than half the length of the hash output to match the birthday attack bound.

4.3. 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.

5. IANA Considerations

TBD

6. Operational Considerations

While efficient, UDP has no build-in congestion-avoidance 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.

7. Security Considerations

The security of the UDP based publication channel depends on the subscription channel. Typically, both NETCONF and RESTCONF support the secure configuration of the private key for the publication channel. So that the message data can be encrypted by using symmetric key algorithms.

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

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <https://www.rfc-editor.org/info/rfc2104>.

[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>.

[RFC2202] Cheng, P. and R. Glenn, "Test Cases for HMAC-MD5 and HMAC-SHA-1", RFC 2202, DOI 10.17487/RFC2202, September 1997, <https://www.rfc-editor.org/info/rfc2202>.

[RFC4231] Nystrom, M., "Identifiers and Test Vectors for HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512", RFC 4231, DOI 10.17487/RFC4231, December 2005, <https://www.rfc-editor.org/info/rfc4231>.

[RFC4493] Song, JH., Poovendran, R., Lee, J., and T. Iwata, "The AES-CMAC Algorithm", RFC 4493, DOI 10.17487/RFC4493, June 2006, <https://www.rfc-editor.org/info/rfc4493>.

[RFC6151] Turner, S. and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", RFC 6151, DOI 10.17487/RFC6151, March 2011, <https://www.rfc-editor.org/info/rfc6151>.

[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-subscribed-notifications]
Voit, E., Clemm, A., Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Custom Subscription to Event Notifications", draft-ietf-netconf-subscribed-notifications-03 (work in progress), July 2017.

[I-D.ietf-netconf-yang-push]
Clemm, A., Voit, E., Prieto, A., Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "Subscribing to YANG datastore push updates", draft-ietf-netconf-yang-push-08 (work in progress), August 2017.

[I-D.voit-netconf-notification-messages]
Voit, E., Bierman, A., Clemm, A., and T. Jenkins, "Notification Message Headers and Bundles", draft-voit-netconf-notification-messages-01 (work in progress), July 2017.

9.3. URIs

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

Appendix A. An Appendix

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