Network Working Group J. Dai
INTERNET-DRAFT X. Wang
Intended Status: Proposed Standard Y. Kou
Expires: December 18, 2022 L. Zhou
China Information Communication Technologies Group
June 18, 2022

Using NETCONF over QUIC connection
draft-dai-netconf-quic-netconf-over-quic-02

Abstract

The Network Configuration Protocol (NETCONF) provides mechanisms to install, manipulate, and delete the configuration of network devices. At present, almost all implementations of NETCONF are based on TCP based protocol. QUIC, a new UDP-based, secure and multiplexed transport protocol, can facilitate to improve the transportation performance, information security and resource utility when being used as an infrastructure layer of NETCONF. This document describes how to use the QUIC protocol as the transport protocol of NETCONF(It is so called NETCONFoQUIC).

Status of this Memo

Copyright and License Notice

Table of Contents

1. Introduction

The Network Configuration Protocol (NETCONF) [RFC6241] defines a mechanism through which the configuration of network devices can be installed, manipulated, and deleted.

NETCONF can be conceptually partitioned into four layers: Content layer, operation layer, message layer and security transport layer.

The Secure Transport layer provides a communication path between the client and server. NETCONF can be layered over any transport protocol that provides a set of basic requirements, the requirements include the following aspects:

(1). NETCONF is connection-oriented, requiring a persistent connection between peers. This connection MUST provide reliable, sequenced data delivery. NETCONF connections are long-lived, persisting between protocol operations.

(2). NETCONF connections MUST provide authentication, data integrity, confidentiality, and replay protection. NETCONF depends on the transport protocol for this capability.

So, the NETCONF protocol is not bound to any particular transport protocol, but allows a mapping to define how it can be implemented over any specific protocol. At the present, there are a few secure transport protocols that can be used to carry NETCONF:

(1). [RFC6242] specifies how to use secure shell(SSH) as the secure transport layer of NETCONF.

(2). [RFC5539] specifies how to use transport layer security(TLS) as the secure transport layer of NETCONF.

(3). [RFC4743] specifies how to use simple object access protocol(SOAP)as the secure transport layer of NETCONF.

(4). [RFC4744] specifies how to use blocks extensible exchange protocol(BEEP) as the secure transport layer of NETCONF.

However, because of the connection-oriented feature, almost all of the current secure transport protocols used by NETCONF is TCP based. As is well known, TCP has some shortcomings such as head-of-line blocking.

[I-D.ietf-quic-transport] specifies a new transport protocol that has the following features:

(1). UDP based

(2). Stream multiplexing

(3). Stream and connection-level flow control

(4). Low-latency connection establishment

(5). Authenticated and encrypted header and payload

It can be learned from the afore-mentioned information that QUIC is also a proper candidate transport protocol for the secure transport layer of NETCONF. In addition, QUIC can perfectly fix the shortcoming such as head of line blocking of TCP. This document specifies how to use QUIC as the secure transport protocol for QUIC.

In this document, the terms "client" and "server" are used to refer to the two ends of the QUIC connection. The client actively initiates the QUIC connection. The terms "manager" and "agent" are used to refer to the two ends of the NETCONF protocol session. The manager issues NETCONF remote procedure call (RPC) commands, and the agent replies to those commands. Generally, a "manager" is a "client" meanwhile an "agent" is a "server".

2. Terminology

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].

3. Connection management

3.1. Draft Version Identification

RFC Editor’s Note: Please remove this section prior to publication of a final version of this document.

NETCONFoQUIC uses the token "NoQ" to identify itself in ALPN and Alt-Svc. Only implementations of the final, published RFC can identify themselves as "NoQ". Until such an RFC exists, implementations MUST NOT identify themselves using this string.

Implementations of draft versions of the protocol MUST add the string "-" and the corresponding draft number to the identifier.

3.2. Connection setup

3.2.1. Version negotiation

QUIC versions are identified using a 32-bit unsigned number, and the version 0x00000000 is reserved to represent version negotiation.

Version negotiation ensures that client and server agree to a QUIC version that is mutually supported.

A server sends a Version Negotiation packet where multiple QUIC versions are listed to the client, the order of the values reflects the server’s preference (with the first value being the most preferred version). Reserved versions MAY be listed, but unreserved versions which are not supported by the alternative SHOULD NOT be present in the list.

When received the Version Negotiation packet, Clients MUST ignore any included versions which they do not support.

If both of the server and the client support the QUIC version that uses TLS version 1.3 or greater as its handshake protocol, the afore-mentioned QUIC version should be the preferred QUIC version of the server and the client.

3.2.2. Connection establishment

QUIC connections are established as described in [I-D.ietf-quic-transport]. During connection establishment, NETCONFoQUIC support is indicated by selecting the ALPN token "NoQ" in the crypto handshake.

The peer acting as the NETCONF manager MUST also act as the client meanwhile the peen acting as the NETCONF agent must also act as the server.

The manager should the initiator of the QUIC connection to the agent meanwhile the agent act as a connection acceptor.

3.3. Connection Closure

3.3.1. QUIC connection termination mode

There are following methods to terminate a QUIC connection:

(1) idle timeout: If the idle timeout is enabled, a connection is silently closed and the state is discarded when it remains idle for longer than both the advertised idle timeout and three times the current Probe Timeout (PTO).

(2) immediate close: An endpoint sends a CONNECTION_CLOSE frame (Section 19.19 of [I-D.ietf-quic-transport]) to terminate the connection immediately.

(3) stateless reset: A stateless reset is provided as an option of last resort for an endpoint that does not have access to the state of a connection.

3.3.2. NETCONFoQUIC consideration for connection termination

When a NETCONF session is implemented based on a QUIC connection, the idle timeout should be disabled in order to keep the QUIC connection persistent even if the NETCONF session is idle.

When a NETCONF server receives a <close-session> request, it will gracefully close the NETCONF session. The server must close the associated QUIC connection.

When a NETCONF entity receives a <kill-session> request for an open session, it should close the associated QUIC connection.

When a NETCONF entity detects any QUIC connection interrupt status, it should send a <close-session> request to the peer NETCONF entity.

When a stateless reset event occurs, nothing needs to be done by either the manager or the agent.

4 Stream mapping and usage

[RFC6241] specifies protocol layers of NETCONF, the protocol layers structure can also be seen from figure 1 of this document, it is noted that this figure is just a citation from [RFC6241].

```
               Layer                 Example
          +-------------+      +-----------------+      +----------------+
      (4) |   Content   |      |  Configuration  |      |  Notification  |
          |             |      |      data       |      |      data      |
          +-------------+      +-----------------+      +----------------+
                 |                       |                      |
          +-------------+      +-----------------+              |
      (3) | Operations  |      |  <edit-config>  |              |
          |             |      |                 |              |
          +-------------+      +-----------------+              |
                 |                       |                      |
          +-------------+      +-----------------+      +----------------+
      (2) |  Messages   |      |     <rpc>,      |      | <notification> |
          |             |      |   <rpc-reply>   |      |                |
          +-------------+      +-----------------+      +----------------+
                 |                       |                      |
          +-------------+      +-----------------------------------------+
      (1) |   Secure    |      |  SSH, TLS, BEEP/TLS, SOAP/HTTP/TLS, ... |
          |  Transport  |      |                                         |
          +-------------+      +-----------------------------------------+
```

Figure 1: NETCONF Protocol Layers

It can be learned from figure 1 that there are two kinds of main data flow exchanged between manager and agent:

(1) Configuration data from manager to agent.

(2) Notification data from agent to manager.

The two kinds of data flow should be mapped into QUIC streams.

QUIC Streams provide a lightweight, ordered byte-stream abstraction to an application. Streams can be unidirectional or bidirectional meanwhile streams can be initiated by either the client or the server. Unidirectional streams carry data in one direction: from the initiator of the stream to its peer. Bidirectional streams allow for data to be sent in both directions.

QUIC uses Stream ID to identify the stream. The least significant bit (0x1) of the stream ID identifies the initiator of the stream. The second least significant bit (0x2) of the stream ID distinguishes between bidirectional streams (with the bit set to 0) and unidirectional streams. Table 1 describes the four types of streams and this table can also be seen from [I-D.ietf-quic-transport].

Bits	Stream Type
0x0	Client-Initiated, Bidirectional
0x1	Server-Initiated, Bidirectional
0x2	Client-Initiated, Unidirectional
0x3	Server-Initiated, Unidirectional

Table 1: Stream ID Types

4.1. Bidirectional stream between manager and agent

The NETCONF protocol uses an RPC-based communication model. So, the configuration data from manager to agent is exchanged based on '<RPC>' (the manager initiating) and '<RPC-Reply>' (sent by the agent) and so on. So the messages used to exchange configuration data should be mapped into one or more bidirectional stream whose stream type is 0.

4.2. Unidirectional stream from agent to manager

There are some notification data exchanged between the agent and the manager. Notification is a server-initiated message indicating that a certain event has been recognized by the server.

Notification messages are initiated by the agent and no reply is needed from the manager. So the messages used to exchange configuration data should be mapped into one unidirectional stream whose stream type is 3.

5 Endpoint Authentication

5.1 using QUIC handshake authentication

NETCONFoQUIC is recommended to use the QUIC version uses TLS version 1.3 or greater. Then, the TLS handshake process can be used for endpoint authentication.

5.1.1. Server Identity

During the TLS negotiation, the client MUST carefully examine the certificate presented by the server to determine if it meets the client's expectations. Particularly, the client MUST check its understanding of the server hostname against the server's identity as presented in the server Certificate message, in order to prevent man- in-the-middle attacks.

Matching is performed according to the rules below (following the example of [RFC4642]):

- The client MUST use the server hostname it used to open the connection (or the hostname specified in the TLS "server_name" extension [RFC5246]) as the value to compare against the server name as expressed in the server certificate. The client MUST NOT use any form of the server hostname derived from an insecure remote source.

- If a subjectAltName extension of type dNSName is present in the certificate, it MUST be used as the source of the server's identity.

- Matching is case-insensitive.

- A "*" wildcard character MAY be used as the leftmost name component in the certificate. For example, *.example.com would match a.example.com, foo.example.com, etc., but would not match example.com.

- If the certificate contains multiple names then a match with any one of the fields is considered acceptable.

If the match fails, the client MUST either ask for explicit user confirmation or terminate the connection and indicate the server's identity is suspect.

Additionally, clients MUST verify the binding between the identity of the servers to which they connect and the public keys presented by those servers. Clients SHOULD implement the algorithm in Section 6 of [RFC5280] for general certificate validation, but MAY supplement that algorithm with other validation methods that achieve equivalent levels of verification (such as comparing the server certificate against a local store of already-verified certificates and identity bindings).

If the client has external information as to the expected identity of the server, the hostname check MAY be omitted.

5.1.2. Client Identity

The server MUST verify the identity of the client with certificate- based authentication according to local policy to ensure that the incoming client request is legitimate before any configuration or state data is sent to or received from the client.

5.2 using third-party authentication

A third-party authentication mechanism can also be used for NETCONFoQUIC endpoint authentication. for example, a SASL profile based authentication method can be used.

6. Security Considerations

The security considerations described throughout [RFC5246] and [RFC4741] apply here as well.

This document in its current version does not support third-party authentication (e.g., backend Authentication, Authorization, and Accounting (AAA) servers) due to the fact that TLS does not specify this way of authentication and that NETCONF depends on the transport protocol for the authentication service. If third-party authentication is needed, BEEP or SSH transport can be used.

An attacker might be able to inject arbitrary NETCONF messages via some application that does not carefully check exchanged messages or deliberately insert the delimiter sequence in a NETCONF message to create a DoS attack. Hence, applications and NETCONF APIs MUST ensure that the delimiter sequence defined in Section 2.1 never appears in NETCONF messages; otherwise, those messages can be dropped, garbled, or misinterpreted. If the delimiter sequence is found in a NETCONF message by the sender side, a robust implementation of this document should warn the user that illegal characters have been discovered. If the delimiter sequence is found in a NETCONF message by the receiver side (including any XML attribute values, XML comments, or processing instructions), a robust implementation of this document must silently discard the message without further processing and then stop the NETCONF session.

Finally, this document does not introduce any new security considerations compared to [RFC4742].

7 IANA Considerations

This document creates a new registration for the identification of NETCONFoQUIC in the "Application Layer Protocol Negotiation (ALPN) Protocol IDs" registry established in [RFC7301].

The "noq" string identifies NETCONFoQUIC:

Protocol: NETCONFoQUIC

Identification Sequence: 0x6e 0x6f 0x71 ("noq")

Specification: This document

In addition, it is requested for IANA to reserve a UDP port TBD for 'NETCONF over QUIC'.

8 Acknowledgements

This document is written by referring [I-D.ietf-quic-transport] edited by Jana Iyengar and Martin Thomson and [I-D.ietf-quic-http] edited by Mike Bishop, and from [RFC7858] authored by Zi Hu, Liang Zhu, John Heidemann, Allison Mankin, Duane Wessels, and Paul Hoffman.

Many thanks to all the afore-mentioned editors and authors.

9 References

9.1 Normative References

[I-D.ietf-quic-transport] Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", draft-ietf-quic-transport-27 (work in progress), Fegruary 2019.

[I-D.ietf-quic-tls] M. Thomson. and S. Turner., " Using TLS to Secure QUIC", draft-ietf-quic-tls-27 (work in progress), February 2019.

[RFC4741] Enns, R., "NETCONF Configuration Protocol", RFC 4741, December 2006.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC6241] Enns, R., "NETCONF Configuration Protocol", RFC 6241, December 2011.

[RFC6242] M. Wasserman., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, June 2011.

[RFC5539] Dierks, T. and E. Rescorla, "NETCONF over Transport Layer Security (TLS)", RFC 5539, May 2009.

[RFC4743] T. Garddard, "Using NETCONF over the Simple Object Access Protocol (SOAP)", RFC 4743, December 2006.

[RFC4744] E. Lear, "Using the NETCONF Protocol over the Blocks Extensible Exchange Protocol (BEEP)", RFC 4743, December 2006.

[I-D.ietf-quic-http] Bishop, M., "Hypertext Transfer Protocol Version 3 (HTTP/3)", draft-ietf-quic-http-18 (work in progress), January 2019.

9.2 Informative References

[RFC5246] M. Badra, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC6101] M. Rose, "The Secure Sockets Layer (SSL) Protocol Version 3.0", RFC 6101, August 2011.

[RFC3080] M. Rose, "The Blocks Extensible Exchange Protocol Core", RFC 3080, March 2001.

Authors' Addresses

Jinyou Dai
China Information Communication Technologies Group.
Gaoxin 4th Road 6#
Wuhan, Hubei 430079
China

Email: djy@fiberhome.com

Xueshun Wang
China Information Communication Technologies Group.
Gaoxin 4th Road 6#
Wuhan, Hubei 430079
China

Email: xswang@fiberhome.com

Yang Kou
China Information Communication Technologies Group.
Gaoxin 4th Road 6#
Wuhan, Hubei 430079
China

Email: ykou@fiberhome.com

Lifen Zhou
China Information Communication Technologies Group.
Gaoxin 4th Road 6#
Wuhan, Hubei 430079
China

Email: lfzhou@fiberhome.com