pce
Internet-Draft
Intended status: Standards Track
Expires: 27 October 2026

F. Yang
China Mobile
C. Lin
New H3C Technologies
S. Sidor
Cisco Systems, Inc.
T. Han
China Mobile

25 April 2026

PCEP over QUIC
draft-yang-pce-pcep-over-quic-04

Abstract

This document specifies the use of QUIC streams to implement the PCEP protocol for efficient and secure data transmission.

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 27 October 2026.

Copyright Notice

Table of Contents

1. Introduction

The Path Computation Element Communication Protocol (PCEP) [RFC5440] facilitates communication between Path Computation Clients (PCCs) and Path Computation Elements (PCEs) to optimize network routing and resource utilization.

QUIC (Quick UDP Internet Connections) [RFC9369] is a transport protocol based on UDP, offering low-latency connections, multiplexing, improved congestion control, and built-in encryption, similar to TLS.

Implementing PCEP over QUIC offers several key benefits:

* Faster Connection Establishment: QUIC's quick handshake reduces connection setup times.

* No Head-of-Line Blocking: Multiple PCEP sessions can run concurrently over a single connection.

* Improved Packet Loss Recovery: QUIC's efficient mechanisms ensure reliable data transmission.

* Enhanced Security: Built-in encryption ensures the confidentiality and integrity of PCEP messages.

* High Resiliency: Support for multiple sessions within a single connection enhances system reliability.

In summary, using QUIC for PCEP enhances efficiency, security, and reliability in network routing communications.

This document primarily describes the protocol adaptations necessary for using a QUIC connection with the PCEP protocol. The protocol process fully adheres to the specifications described in RFC 5440.

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. Terminology

PCEPoQ: PCEP using QUIC.

2.1. Establish PCEP/QUIC Connection

The process of establishing a PCEP session is described in [RFC5440].

Before two PCEPoQ speakers start exchanging routing information, they must establish a PCEP session. It is established in two phases:

First, a QUIC connection is established at the transport layer as described in [RFC9000]. When setting up a QUIC connection, PCEPoQ uses UDP port number TBD1 and employs TLS for security. During the TLS handshake, the Application-Layer Protocol Negotiation (ALPN) token "pcepoq" is used.

Second, Establish a PCEPoQ session over this transport connection. PCC over QUIC acts as the QUIC client, and PCE over QUIC serves as the QUIC server.

2.2. Establish PCEP/QUIC Control Channel

After PCEPoQ session established, the PCEPoQ speaker establishes a bidirectional stream for the "PCEPoQ control channel." The control channel is used to establish a PCEPoQ relationship between the PCC and the PCE over QUIC. OPEN messages are used to establish the PCEPoQ control channel. After the channel is established, KeepAlive messages are sent to maintain the control channel. Close messages are used to terminate the control channel. Notify messages are used to send specified notifications. When an error occurs, a PCEErr message is sent through the PCEPoQ control channel to notify the PCEPoQ peer.

2.3. Establish PCEP/QUIC data Channel

After establishing the control channel, each PCEPoQ speaker may create data channels using unidirectional QUIC streams. These data channels are used to carry PCEReq and PCERep messages. For IPv4 and IPv6 address families, separate data channels can also be established. The advantage of separating the control channel from the data channels is that it allows for higher priority handling of the control channel’s traffic, ensuring timely processing of control messages without delay caused by handling data messages. Moreover, if PCEPoQ supports other address families in the future, using separate data channels can ensure that different address families do not interfere with each other.

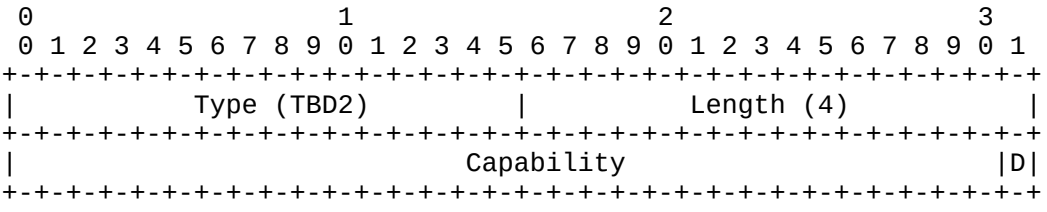
For different PCEPoQ sessions, the same QUIC connection is used. The "pcepoq" ALPN token is not session-specific.

3. Protocol Definitions

3.1. PCEP Over QUIC Capability

Once a QUIC connection is established, the PCC and PCE over QUIC initiate the PCEPoQ session establishment process, during which various session parameters are negotiated. These parameters are carried within Open messages and include the Keepalive timer, DeadTimer, and policy rules that specify the conditions under which path computation requests may be sent to the PCE, as well as the supported PCEPoQ capabilities and other detailed capabilities. If the PCEPoQ session establishment phase fails because the PCEPoQ peers disagree on the session parameters or one of the PCEPoQ peers does not respond after the expiration of the establishment timer, successive retries are permitted. However, an implementation should use an exponential back-off procedure for session establishment retries.

To negotiate PCEPoQ-related capabilities, this document extends the PCEPoQ Capability TLV Figure 1.



Type : 2 bytes (TBD2)
Length : 2 bytes (4)
Capability:
D: Bit 0, support data channel, this document

Figure 1: PCEPoQ Capability TLV

When sending an OPEN message to establish a PCEPoQ session, the local PCEPoQ capabilities are included. If the capability negotiation fails, the PCEPoQ session establishment fails. After a failure, After a failure, if no other sessions exist, a QUIC connection may choose to remain open or close based on specific implementation, though this document does not elaborate further.

3.2. Maintenance of control channel

Once a PCEPoQ session has been established, a PCE or PCC over QUIC may want to know that its PCEPoQ peer is still available for use.

PCEPoQ includes a keepalive mechanism based on a Keepalive timer, a DeadTimer, and a Keepalive message. The PCEP KeepAlive message is sent through the PCEPoQ control channel.

If no KeepAlive message is received within the DeadTimer period, the PCEPoQ session should be reset and a new session should be attempted. While resetting the PCEPoQ session, the corresponding PCEPoQ data channel should also be reset.

The Notification message can be sent by either a PCE to a PCC, or by a PCC to a PCE, to notify of a specific event. This message is transmitted through the PCEPoQ control channel.

There are several circumstances in which a PCE over QUIC may want to notify a PCC over QUIC of a specific event. For example, suppose that the PCE over QUIC suddenly gets overloaded, potentially leading to unacceptable response times.

The PCE over QUIC may want to notify one or more PCCs over QUIC that some of their requests (listed in the notification) will not be satisfied or may experience unacceptable delays. Upon receiving such notification, the PCC over QUIC may decide to redirect its path computation requests to another PCE over QUIC should an alternate PCE be available. Similarly, a PCC over QUIC may desire to notify a PCE over QUIC of a particular event such as the cancellation of pending requests.

3.3. PCEPoQ Connection Termination

When one of the PCEP over QUIC peers wishes to terminate a PCEPoQ session, it first sends a Close message. This message is transmitted through the PCEPoQ control channel. The sender then closes the PCEPoQ session. Upon receiving the Close Message, the recipient closes the corresponding PCEPoQ session.

If there are no other PCEPoQ sessions, the QUIC connection can be closed or kept permanently active depending on the implementation. This document assumes that the QUIC connection remains open, and the following sections on the state machine are based on this assumption and will not repeat it.

The PCEP over QUIC Error message is sent in several situations: when a protocol error condition is met or when the request is not compliant with the PCEP over QUIC specification (e.g., capability not supported, reception of a message with a mandatory missing object, policy violation, unexpected message, unknown request reference).

This message is transmitted through the PCEPoQ control channel.

3.4. PCEPoQ Framing Layer

Some PCEPoQ messages, although sent in the control channel, are meant for a function channel, such as the responding OPEN message or KEEPALIVE message for a function channel. These messages need to carry the corresponding function channel/stream ID information.

There are two types of PCEPoQ Frames: Control Data and Data.

Data frames have the following format:

```
PCEPoQ Data Frame Format {
   Type (16) = 0,
   Length (16),
   Frame Payload (...)
}

PCEPoQ Control Data Frame Format {
  Type (16) = 1,
  Length (16),
  Stream ID (62),
  padding (2) = 0,
  Frame Payload (...)
}
```

Type: two octets, identifying the frame type.

Length: The two-byte unsigned integer that describes the length in bytes of the frame payload.

Stream ID: A 62-bit integer indicating the receiving stream ID of this message.

Frame Payload: PCEP messages.

The following table Table 1 lists the frame type to be used when BGP messages are sent in different channels.

	Control Channel	Function Channel
OPEN	Control	Data
KEEPALIVE	Control	Data
NOTIFICATION	Control	Data
PCErr	Control	Data
PCReq	/	Data
PCRep	/	Data
Close	Control	Data

Table 1: PCEPoQ Frame Type Mapping

3.5. Path Computation Request and Reply

Once a PCC has successfully established a PCEPoQ session with one or more PCEs, if an event is triggered that requires the computation of a set of paths, the PCC first selects one or more PCEs. Once the PCC has selected a PCE, it sends a path computation request to the PCE (PCReq message) by PCEPoQ connection.

The PCReq message can only be sent and received through the PCEPoQ data channel. If this message is received on the control channel, it will be ignored.

Upon receiving a path computation request from a PCC, the PCE initiates a path computation. The result is then sent to the PCC via a PCRep message through the PCEPoQ data channel.

The PCRep message can only be sent and received through the PCEPoQ data channel. If this message is received on the control channel, it will be ignored.

4. PCEPoQ Finite State Machine (FSM)

4.1. Events for the PCEPoQ FSM

4.1.1. Optional Events Linked to Optional Session Attributes

The Inputs to the PCEPoQ FSM are events. Events can either be mandatory or optional. Some optional events are linked to optional session attributes. Optional session attributes enable several groups of FSM functionality.

The linkage between FSM functionality, events, and the optional session attributes are described below.

Group 1: Automatic Administrative Events (Start/Stop)

Optional Session Attributes: AllowAutomaticStart,
AllowAutomaticStop,
DampPeerOscillations,
IdleHoldTime,
IdleHoldTimer

Group 2: Unconfigured Peers

Optional Session Attributes: AcceptConnectionsUnconfiguredPeers

Group 3: QUIC processing

Optional Session Attributes: PassiveQuicEstablishment,
TrackQuicState

Group 4: PCEPoQ Message Processing

Optional Session Attributes: DelayOpen,
DelayOpenTime,
DelayOpenTimer,
SendNOTIFICATIONwithoutOPEN,
CollisionDetectEstablishedState

4.1.2. Administrative Events

Event 1: ManualStart

Event 2: ManualStop

Event 3: AutomaticStart

Event 4: ManualStart_with_PassiveQuicEstablishment

Event 5: AutomaticStart_with_PassiveQuicEstablishment

Event 6: AutomaticStart_with_DampPeerOscillations

Event 7:
AutomaticStart_with_DampPeerOscillations_and_PassiveQUICEstablishment

Event 8: AutomaticStop

4.1.3. Timer Events

Event 9: ConnectRetryTimer_Expires

Event 10: HoldTimer_Expires

Event 11: KeepaliveTimer_Expires

Event 12: DelayOpenTimer_Expires

Event 13: IdleHoldTimer_Expires

4.1.4. QUIC Connection-Based Events

Event 14: QuicConnection_Valid

Event 15: QUIC_CR_Invalid

Event 16: QUIC_CR_Acked

Event 17: QuicConnectionConfirmed

Event 18: QuicConnectionFails

4.1.5. PCEPoQ Message-Based Events

Event 19: PCEPOpen

Event 20: PCEPOpen with DelayOpenTimer running

Event 21: PCEPHeaderErr

Event 22: PCEPOpenMsgErr

Event 23: OpenCollisionDump

Event 24: NotifMsgVerErr

Event 25: NotifMsg

Event 26: KeepAliveMsg

Event 27: NotificationMsg

Event 28: ErrorMsg

4.2. Description of FSM

The section describes the PCEPoQ finite state machine (FSM).

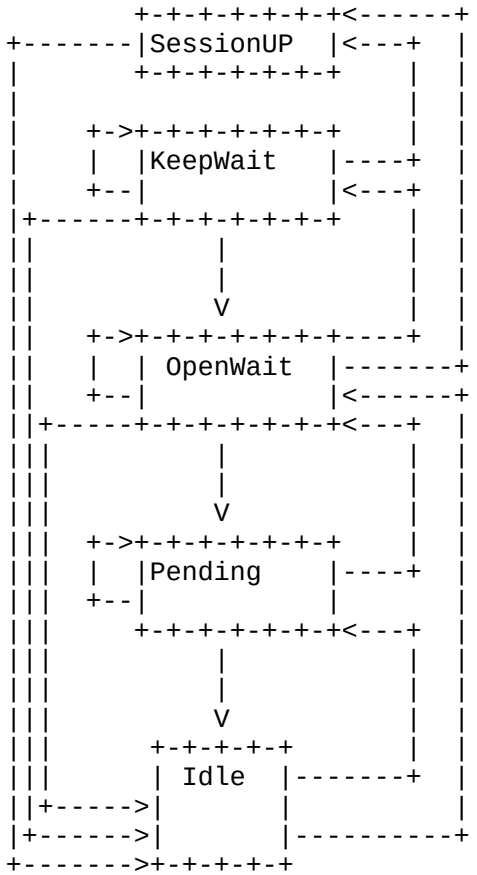


Figure 2: PCEPoQ Finite State Machine

PCEPoQ defines the following set of variables:

Connect: the timer (in seconds) started after having initialized a QUIC connection using the PCEPoQ-registered UDP port. The value of the Connect timer is 60 seconds.

ConnectRetry: the number of times the system has tried to establish a QUIC connection with a PCEPoQ peer without success.

ConnectMaxRetry: the maximum number of times the system tries to establish a QUIC connection using the PCEPoQ-registered UDP port before going back to the Idle state. The value of the ConnectMaxRetry is 5.

OpenWait: the timer that corresponds to the amount of time a PCEPoQ peer will wait to receive an PCEPoQ Open message from the PCEPoQ peer after the expiration of which the system releases the PCEPoQ resource and goes back to the Idle state. The OpenWait timer has a fixed value of 60 seconds.

KeepWait: the timer that corresponds to the amount of time a PCEPoQ peer will wait to receive a Keepalive or a PCEoQErr message from the PCEPoQ peer after the expiration of which the system releases the PCEPoQ resource and goes back to the Idle state. The KeepWait timer has a fixed value of 60 seconds.

OpenRetry: the number of times the system has received an PCEPoQ Open message with unacceptable PCEPoQ session characteristics.

The following two state variables are defined:

RemoteOK: a boolean that is set to 1 if the system has received an acceptable PCEPoQ Open message.

LocalOK: a boolean that is set to 1 if the system has received a PCEPoQ Keepalive message acknowledging that the PCEPoQ Open message sent to the peer was valid.

4.2.1. Idle State:

The idle state is the initial PCEPoQ state where the PCEPoQ (also referred to as "the system") waits for an initialization event that can either be manually triggered by the user(configuration) or automatically triggered by various events. In Idle state, PCEPoQ resources are allocated (memory, potential process, etc.) but no PCEPoQ messages are accepted from any PCEPoQ peer.

The following set of variables are initialized:

* QUICRetry=0,
* LocalOK=0,
* RemoteOK=0,
* OpenRetry=0.

In this state, PCEP FSM refuses all incoming PCEP connections for this peer. No resources are allocated to the peer. In response to a ManualStart event (Event 1) or an AutomaticStart event (Event 3), the local system:

* Initiates a QUIC connection with the PCEP peer,

* starts the ConnectRetryTimer with the initial value,

* changes its state to Pending.

The ManualStop event (Event 2) and AutomaticStop (Event 8) event are ignored in the Idle state.

In response to a ManualStart_with_PassiveQuicEstablishment event (Event 4) or AutomaticStart_with_PassiveQuicEstablishment event (Event 5), the local system:

* Initiates a QUIC connection with the PCEP peer,

* sets the ConnectRetryCounter to zero,

* starts the ConnectRetryTimer with the initial value,

* changes its state to OpenWait.

If the DampPeerOscillations attribute is set to TRUE, the following three additional events may occur within the Idle state:

* AutomaticStart_with_DampPeerOscillations (Event 6),

* AutomaticStart_with_DampPeerOscillations_and_PassiveQuicEstablishment (Event 7),

* IdleHoldTimer_Expires (Event 13).

Upon receiving these 3 events, the local system will use these events to prevent peer oscillations. The method of preventing persistent peer oscillation is outside the scope of this document.

Any other event (Events 9-12, 15-28) received in the Idle state does not cause change in the state of the local system.

4.2.2. Pending State:

In this state, PCEP FSM is waiting for the QUIC connection to be completed.

The start events (Events 1, 3-7) are ignored in the Pending state.

In response to a ManualStop event (Event 2), the local system:

* drops the QUIC connection,

* releases all PCEP resources,

* sets ConnectRetryCounter to zero,

* stops the ConnectRetryTimer and sets ConnectRetryTimer to zero, and

* changes its state to Idle.

In response to the ConnectRetryTimer_Expires event (Event 9), the local system:

* drops the QUIC connection,

* restarts the ConnectRetryTimer,

* stops the DelayOpenTimer and resets the timer to zero,

* initiates a QUIC connection to the other PCEP peer,

* continues to listen for a connection that may be initiated by the remote PCEP peer, and

* stays in the Pending state.

If the DelayOpenTimer_Expires event (Event 12) occurs in the Connect state, the local system:

* sends an OPEN message to its peer,

* sets the HoldTimer to a large value, and

* changes its state to OpenWait.

If the PCEP FSM receives a QuicConnection_Valid event (Event 14), the QUIC connection is processed, and the connection remains in the Pending state.

If the PCEP FSM receives a QUIC_CR_Invalid event (Event 15), the local system rejects the QUIC connection, and the connection remains in the Pending state.

If the QUIC connection succeeds (Event 16 or Event 17), the local system checks the DelayOpen attribute prior to processing. If the DelayOpen attribute is set to TRUE, the local system:

* stops the ConnectRetryTimer (if running) and sets the ConnectRetryTimer to zero,

* sets the DelayOpenTimer to the initial value, and

* stays in the Pending state.

If the DelayOpen attribute is set to FALSE, the local system:

* stops the ConnectRetryTimer (if running) and sets the ConnectRetryTimer to zero,

* completes PCEP initialization

* sends an OPEN message to its peer,

* sets the HoldTimer to a large value, and

* changes its state to OpenWait.

A HoldTimer value of 4 minutes is suggested.

If the QUIC connection fails (Event 18), the local system checks the DelayOpenTimer. If the DelayOpenTimer is running, the local system:

* restarts the ConnectRetryTimer with the initial value,

* stops the DelayOpenTimer and resets its value to zero,

* continues to listen for a connection that may be initiated by the remote PCEP peer, and

* Remain its state Pending.

If the DelayOpenTimer is not running, the local system:

* stops the ConnectRetryTimer to zero,

* drops the QUIC connection,

* releases all PCEP resources, and

* changes its state to Idle.

If an OPEN message is received while the DelayOpenTimer is running (Event 20), the local system:

* stops the ConnectRetryTimer (if running) and sets the ConnectRetryTimer to zero,

* completes the PCEP initialization,

* stops and clears the DelayOpenTimer (sets the value to zero),

* sends an OPEN message,

* sends a KEEPALIVE message,

* and changes its state to KeepWait.

If the value of the autonomous system field is the same as the local Autonomous System number, set the connection status to an internal connection; otherwise it will be "external".

If PCEP message header checking (Event 21) or OPEN message checking detects an error (Event 22), the local system:

* (optionally) If the SendNOTIFICATIONwithoutOPEN attribute is set to TRUE, then the local system first sends a NOTIFICATION message with the appropriate error code, and then

* stops the ConnectRetryTimer (if running) and sets the ConnectRetryTimer to zero,

* releases all PCEP resources,

* drops the QUIC connection,

* increments the ConnectRetryCounter by 1,

* (optionally) performs peer oscillation damping if the DampPeerOscillations attribute is set to TRUE, and

* changes its state to Idle.

If a PCErr message is received with a version error (Event 24), the local system checks the DelayOpenTimer. If the DelayOpenTimer is running, the local system:

* stops the ConnectRetryTimer (if running) and sets the ConnectRetryTimer to zero,

* stops and resets the DelayOpenTimer (sets to zero),

* releases all PCEP resources,

* drops the QUIC connection, and

* changes its state to Idle.

If the DelayOpenTimer is not running, the local system:

* stops the ConnectRetryTimer and sets the ConnectRetryTimer to zero,

* releases all PCEP resources,

* drops the QUIC connection,

* increments the ConnectRetryCounter by 1,

* performs peer oscillation damping if the DampPeerOscillations attribute is set to True, and

* changes its state to Idle.

In response to any other events (Events 8, 10-11, 13, 19, 23, 25-28), the local system:

* if the ConnectRetryTimer is running, stops and resets the ConnectRetryTimer (sets to zero),

* if the DelayOpenTimer is running, stops and resets the DelayOpenTimer (sets to zero),

* releases all PCEP resources,

* drops the QUIC connection,

* increments the ConnectRetryCounter by 1,

* performs peer oscillation damping if the DampPeerOscillations attribute is set to True, and

* changes its state to Idle.

The start events (Events 1, 3-7) are ignored in the Pending state.

In response to a ManualStop event (Event 2), the local system:

* If the DelayOpenTimer is running and the SendNOTIFICATIONwithoutOPEN session attribute is set, the local system sends a PCErr with a Cease,

* releases all PCEP resources including stopping the DelayOpenTimer

* drops the QUIC connection,

* sets ConnectRetryCounter to zero,

* stops the ConnectRetryTimer and sets the ConnectRetryTimer to zero, and

* changes its state to Idle.

In response to a ConnectRetryTimer_Expires event (Event 9), the local system:

* restarts the ConnectRetryTimer (with initial value),

* initiates a QUIC connection to the other PCEP peer,

* continues to listen for a TCP connection that may be initiated by a remote PCEP peer, and

* changes its state to OpenWait.

If the local system receives a DelayOpenTimer_Expires event (Event 12), the local system:

* sets the ConnectRetryTimer to zero,

* stops and clears the DelayOpenTimer (set to zero),

* completes the PCEP initialization,

* sends the OPEN message to its remote peer,

* sets its hold timer to a large value, and

* changes its state to OpenWait.

A HoldTimer value of 4 minutes is also suggested for this state transition.

If the local system receives a QuicConnection_Valid event (Event 14), the local system processes the TCP connection flags and stays in the Pending state.

If the local system receives a QUIC_CR_Invalid event (Event 15), the local system rejects the QUIC connection and stays in the Pending State.

In response to the success of a QUIC connection (Event 16 or Event 17), the local system checks the DelayOpen optional attribute prior to processing.

If the DelayOpen attribute is set to TRUE, the local system:

* stops the ConnectRetryTimer and sets the ConnectRetryTimer to zero,

* sets the DelayOpenTimer to the initial value (DelayOpenTime), and

* stays in the Pending state.

If the DelayOpen attribute is set to FALSE, the local system:

* sets the ConnectRetryTimer to zero,

* completes the PCEP initialization,

* sends the OPEN message to its peer,

* sets its HoldTimer to a large value, and

* changes its state to OpenWait.

A HoldTimer value of 4 minutes is suggested as a "large value" for the HoldTimer.

If the local system receives a QUICConnectionFails event (Event 18), the local system:

* restarts the ConnectRetryTimer (with the initial value),

* stops and clears the DelayOpenTimer (sets the value to zero),

* releases all PCEP resource,

* increments the ConnectRetryCounter by 1,

* optionally performs peer oscillation damping if the DampPeerOscillations attribute is set to TRUE, and

* changes its state to Idle.

If an OPEN message is received and the DelayOpenTimer is running (Event 20), the local system:

* stops the ConnectRetryTimer (if running) and sets the ConnectRetryTimer to zero,

* stops and clears the DelayOpenTimer (sets to zero),

* completes the PCEP initialization,

* sends an OPEN message,

* sends a KEEPALIVE message,

* if the HoldTimer value is non-zero,

* starts the KeepaliveTimer to initial value,

* resets the HoldTimer to the negotiated value, else if the HoldTimer is zero

* resets the KeepaliveTimer (set to zero),

* resets the HoldTimer to zero, and

* changes its state to KeepWait.

If the value of the autonomous system field is the same as the local Autonomous System number, set the connection status to an internal connection; otherwise it will be external.

If PCEP message header checking (Event 21) or OPEN message checking detects an error (Event 22), the local system:

* (optionally) sends a PCErr message with the appropriate error code if the SendNOTIFICATIONwithoutOPEN attribute is set to TRUE,

* sets the ConnectRetryTimer to zero,

* releases all PCEP resources,

* drops the QUIC connection,

* increments the ConnectRetryCounter by 1,

* (optionally) performs peer oscillation damping if the DampPeerOscillations attribute is set to TRUE, and

* changes its state to Idle.

If a PCErr message is received with a version error (Event 24), the local system checks the DelayOpenTimer. If the DelayOpenTimer is running, the local system:

* stops the ConnectRetryTimer (if running) and sets the ConnectRetryTimer to zero,

* stops and resets the DelayOpenTimer (sets to zero),

* releases all PCEP resources,

* drops the QUIC connection, and

* changes its state to Idle.

If the DelayOpenTimer is not running, the local system:

* sets the ConnectRetryTimer to zero,

* releases all PCEP resources,

* drops the QUIC connection,

* increments the ConnectRetryCounter by 1,

* (optionally) performs peer oscillation damping if the DampPeerOscillations attribute is set to TRUE, and

* changes its state to Idle.

In response to any other event (Events 8, 10-11, 13, 19, 23, 25-28), the local system:

* sets the ConnectRetryTimer to zero,

* releases all PCEP resources,

* drops the QUIC connection,

* increments the ConnectRetryCounter by one,

* (optionally) performs peer oscillation damping if the DampPeerOscillations attribute is set to TRUE, and

* changes its state to Idle.

4.2.3. OpenWait:

In this state, PCEP FSM waits for an OPEN message from its peer. The start events (Events 1, 3-7) are ignored in the OpenWait state.

If a ManualStop event (Event 2) is issued in the OpenWait state, the local system:

* sends the Close with a Cease,
* sets the ConnectRetryTimer to zero,
* releases all PCEP resources,
* drops the QUIC connection,
* sets the ConnectRetryCounter to zero, and
* changes its state to Idle.

If an AutomaticStop event (Event 8) is issued in the OpenWait state, the local system:

* sends the Close with a Cease,
* sets the ConnectRetryTimer to zero,
* releases all the PCEP resources,
* drops the QUIC connection,
* increments the ConnectRetryCounter by 1,
* (optionally) performs peer oscillation damping if the DampPeerOscillations attribute is set to TRUE, and
* changes its state to Idle.

If the HoldTimer_Expires (Event 10), the local system:

* sends a PCErr message with the error code Hold Timer Expired,
* sets the ConnectRetryTimer to zero,
* releases all PCEP resources,
* drops the QUIC connection,

* increments the ConnectRetryCounter,

* (optionally) performs peer oscillation damping if the DampPeerOscillations attribute is set to TRUE, and

* changes its state to Idle.

If a QUICConnection_Valid (Event 14), QUIC_CR_Acked (Event 16), or a QUICConnectionConfirmed event (Event 17) is received, a second QUIC connection may be in progress.

This second QUIC connection is tracked per Connection Collision processing until an OPEN message is received.

A QUIC Connection Request for an Invalid port (QUIC_CR_Invalid (Event 15)) is ignored.

If a QUICConnectionFails event (Event 18) is received, the local system:

* closes the PCEP connection,

* restarts the ConnectRetryTimer,

* continues to listen for a connection that may be initiated by the remote PCEP peer, and

* changes its state to Pending.

When an OPEN message is received, all fields are checked for correctness. If there are no errors in the OPEN message (Event 19), the local system:

* resets the DelayOpenTimer to zero,

* sets the PCEP ConnectRetryTimer to zero,

* sends a KEEPALIVE message, and

* sets a KeepaliveTimer (via the text below)

* sets the HoldTimer according to the negotiated value,

* changes its state to KeepWait.

If the negotiated hold time value is zero, then the HoldTimer and KeepaliveTimer are not started. If the value of the Autonomous System field is the same as the local Autonomous System number, then

the connection is an "internal" connection; otherwise, it is an "external" connection. If the PCEP message header checking (Event 21) or OPEN message checking detects an error (Event 22), the local system:

* sends a PCErr message with the appropriate error code,

* sets the ConnectRetryTimer to zero,

* releases all PCEP resources,

* drops the QUIC connection,

* increments the ConnectRetryCounter by 1,

* (optionally) performs peer oscillation damping if the DampPeerOscillations attribute is TRUE, and

* changes its state to Idle.

Collision detection mechanisms need to be applied when a valid PCEP OPEN message is received (Event 19 or Event 20). A CollisionDetectDump event occurs when the PCEP implementation determines, by means outside the scope of this document, that a connection collision has occurred. If a connection in the OpenWait state is determined to be the connection that must be closed, an OpenCollisionDump (Event 23) is signaled to the state machine. If such an event is received in the OpenWait state, the local system:

* sends a PCErr with a Cease,

* sets the ConnectRetryTimer to zero,

* releases all PCEP resources,

* drops the QUIC connection,

* increments the ConnectRetryCounter by 1,

* (optionally) performs peer oscillation damping if the DampPeerOscillations attribute is set to TRUE, and

* changes its state to Idle.

If a PCErr message is received with a version error (Event 24), the local system:

* sets the ConnectRetryTimer to zero,

* releases all PCEP resources,

* drops the QUIC connection, and

* changes its state to Idle.

In response to any other event (Events 9, 11-13, 20, 25-28), the local system:

* sends the PCErr with the Error Code Finite State Machine Error,

* sets the ConnectRetryTimer to zero,

* releases all PCEP resources,

* drops the QUIC connection,

* increments the ConnectRetryCounter by 1,

* (optionally) performs peer oscillation damping if the DampPeerOscillations attribute is set to TRUE, and

* changes its state to Idle.

4.2.4. KeepWait State:

In this state, PCEP waits for a KEEPALIVE or PCErr message.

Any start event (Events 1, 3-7) is ignored in the KeepWait state.

In response to a ManualStop event (Event 2) initiated by the operator, the local system:

* sends the PCErr message with a Cease,

* releases all PCEP resources,

* drops the QUIC connection,

* sets the ConnectRetryCounter to zero,

* sets the ConnectRetryTimer to zero, and

* changes its state to Idle.

In response to the AutomaticStop event initiated by the system (Event 8), the local system:

* sends the PCErr message with a Cease,

* sets the ConnectRetryTimer to zero,

* releases all PCEP resources,

* drops the QUIC connection,

* increments the ConnectRetryCounter by 1,

* (optionally) performs peer oscillation damping if the DampPeerOscillations attribute is set to TRUE, and

* changes its state to Idle.

If the HoldTimer_Expires event (Event 10) occurs before a KEEPALIVE message is received, the local system:

* sends the PCErr message with the Error Code Hold Timer Expired,

* sets the ConnectRetryTimer to zero,

* releases all PCEP resources,

* drops the QUIC connection,

* increments the ConnectRetryCounter by 1,

* (optionally) performs peer oscillation damping if the DampPeerOscillations attribute is set to TRUE, and

* changes its state to Idle.

If the local system receives a KeepaliveTimer_Expires event (Event 11), the local system:

* sends a KEEPALIVE message,

* restarts the KeepaliveTimer, and

* remains in the KeepWait state.

In the event of a QUICConnection_Valid event (Event 14), or the success of a TCP connection (Event 16 or Event 17) while in KeepWait, the local system needs to track the second connection.

If a QUIC connection is attempted with an invalid port (Event 15), the local system will ignore the second connection attempt.

If the local system receives a QUICConnectionFails event (Event 18) from the underlying QUIC or a PCErr message (Event 25), the local system:

* sets the ConnectRetryTimer to zero,

* releases all PCEP resources,

* drops the QUIC connection,

* increments the ConnectRetryCounter by 1,

* (optionally) performs peer oscillation damping if the DampPeerOscillations attribute is set to TRUE, and

* changes its state to Idle.

If the local system receives a PCErr message with a version error (NotifMsgVerErr (Event 24)), the local system:

* sets the ConnectRetryTimer to zero,

* releases all PCEP resources,

* drops the QUIC connection, and

* changes its state to Idle.

If the local system receives a valid OPEN message (PCEPOpen (Event 19)), the collision detect function is processed If this connection is to be dropped due to connection collision, the local system:

* sends a PCErr with a Cease,

* sets the ConnectRetryTimer to zero,

* releases all PCEP resources,

* drops the QUIC connection,

* increments the ConnectRetryCounter by 1,

* (optionally) performs peer oscillation damping if the DampPeerOscillations attribute is set to TRUE, and

* changes its state to Idle.

If an OPEN message is received, all fields are checked for correctness. If the PCEP message header checking (PCEPHeaderErr (Event 21)) or OPEN message checking detects an error (PCEPOpenMsgErr (Event 22)), the local system:

* sends a PCErr message with the appropriate error code,

* sets the ConnectRetryTimer to zero,

* releases all PCEP resources,

* drops the QUIC connection,

* increments the ConnectRetryCounter by 1,

* (optionally) performs peer oscillation damping if the DampPeerOscillations attribute is set to TRUE, and

* changes its state to Idle.

If, during the processing of another OPEN message, the PCEP implementation determines, by a means outside the scope of this document, that a connection collision has occurred and this connection is to be closed, the local system will issue an OpenCollisionDump event (Event 23). When the local system receives an OpenCollisionDump event (Event 23), the local system:

* sends a PCErr with a Cease,

* sets the ConnectRetryTimer to zero,

* releases all PCEP resources

* drops the QUIC connection,

* increments the ConnectRetryCounter by 1,

* (optionally) performs peer oscillation damping if the DampPeerOscillations attribute is set to TRUE, and

* changes its state to Idle.

If the local system receives a KEEPALIVE message (KeepAliveMsg(Event 26)), the local system:

* restarts the HoldTimer and

* changes its state to SessionUP.

In response to any other event (Events 9, 12-13, 20, 27-28), the local system:

* sends a PCErr with a code of Finite State Machine Error,

* sets the ConnectRetryTimer to zero,

* releases all PCEP resources,

* drops the QUIC connection,

* increments the ConnectRetryCounter by 1,

* (optionally) performs peer oscillation damping if the DampPeerOscillations attribute is set to TRUE, and

* changes its state to Idle.

4.2.5. SessionUP State:

In the SessionUP state, the PCEP FSM can exchange PCReq, PCRep, PCErr, and KEEPALIVE messages with its peer.

Any Start event (Events 1, 3-7) is ignored in the SessionUP state. In response to a ManualStop event (initiated by an operator) (Event 2), the local system:

* sends the Close message with a Cease,

* sets the ConnectRetryTimer to zero,

* deletes all routes associated with this connection,

* releases PCEP resources,

* drops the QUIC connection,

* sets the ConnectRetryCounter to zero, and

* changes its state to Idle.

In response to an AutomaticStop event (Event 8), the local system:

* sends a PCErr with a Cease,

* sets the ConnectRetryTimer to zero

* deletes all routes associated with this connection,

* releases all PCEP resources,

* drops the QUIC connection,

* increments the ConnectRetryCounter by 1,

* (optionally) performs peer oscillation damping if the DampPeerOscillations attribute is set to TRUE, and

* changes its state to Idle.

One reason for an AutomaticStop event is: A PCEP receives an PCNtf messages with a number of prefixes for a given peer such that the total prefixes received exceeds the maximum number of prefixes configured. The local system automatically disconnects the peer.

If the HoldTimer_Expires event occurs (Event 10), the local system:

* sends a PCErr message with the Error Code Hold Timer Expired,

* sets the ConnectRetryTimer to zero,

* releases all PCEP resources,

* drops the QUIC connection,

* increments the ConnectRetryCounter by 1,

* (optionally) performs peer oscillation damping if the DampPeerOscillations attribute is set to TRUE, and

* changes its state to Idle.

If the KeepaliveTimer_Expires event occurs (Event 11), the local system:

* sends a KEEPALIVE message, and

* restarts its KeepaliveTimer, unless the negotiated HoldTime value is zero.

Each time the local system sends a KEEPALIVE or PCNtf message, it restarts its KeepaliveTimer, unless the negotiated HoldTime value is zero.

A QUICConnection_Valid (Event 14), received for a valid port, will cause the second connection to be tracked.

An invalid QUIC connection (QUIC_CR_Invalid event (Event 15)) will be ignored.

In response to an indication that the QUIC connection is successfully established (Event 16 or Event 17), the second connection SHALL be tracked until it sends an OPEN message.

If a valid OPEN message (PCEPOpen (Event 19)) is received, and if the CollisionDetect SessionUP State optional attribute is TRUE, the OPEN message will be checked to see if it collides with any other connection. If the PCEP implementation determines that this connection needs to be terminated, it will process an OpenCollisionDump event (Event 23). If this connection needs to be terminated, the local system:

* sends a PCErr with a Cease,

* sets the ConnectRetryTimer to zero,

* deletes all routes associated with this connection,

* releases all PCEP resources,

* drops the QUIC connection,

* increments the ConnectRetryCounter by 1,

* (optionally) performs peer oscillation damping if the DampPeerOscillations is set to TRUE, and

* changes its state to Idle.

If the local system receives a PCErr message (Event 24 or Event 25) or a QUICConnectionFails (Event 18) from the underlying TCP, the local system:

* sets the ConnectRetryTimer to zero,

* deletes all routes associated with this connection,

* releases all the PCEP resources,

* drops the QUIC connection,

* increments the ConnectRetryCounter by 1,

* changes its state to Idle.

If the local system receives a KEEPALIVE message (Event 26), the local system:

* restarts its HoldTimer, if the negotiated HoldTime value is non-zero, and

* remains in the SessionUP state.

If the local system receives an PCNtf message (Event 27), the local system:

* processes the message,

* restarts its HoldTimer, if the negotiated HoldTime value is non-zero, and

* remains in the SessionUP state.

If the local system receives an PCNtf message, and the PCNtf message error handling procedure detects an error (Event 28), the local system:

* sends a PCErr message with an error,

* sets the ConnectRetryTimer to zero,

* deletes all routes associated with this connection,

* releases all PCEP resources,

* drops the QUIC connection,

* increments the ConnectRetryCounter by 1,

* (optionally) performs peer oscillation damping if the DampPeerOscillations attribute is set to TRUE, and

* changes its state to Idle.

In response to any other event (Events 9, 12-13, 20-22), the local system:

* sends a PCErr message with the Error Code Finite State Machine Error,

* deletes all routes associated with this connection,

* sets the ConnectRetryTimer to zero,

* releases all PCEP resources,

* drops the QUIC connection,

* increments the ConnectRetryCounter by 1,

* (optionally) performs peer oscillation damping if the DampPeerOscillations attribute is set to TRUE, and

* changes its state to Idle.

5. Stream multiplexing

QUIC offers stream multiplexing but does not provide a mechanism for exchanging stream priority information. Instead, it relies on receiving priority information from the application. PCEPoQ separates streams into control and data channels, assigning different priorities to each, ensuring that control messages receive higher priority processing.

6. Network Migration

QUIC connections are not strictly bound to a single network path. Connection migration uses connection identifiers to allow connections to transfer to a new network path. In the current version of QUIC, only clients can migrate. The PCEPoQ protocol inherits this feature, allowing PCC over QUIC to conduct network migration. PCE over QUIC should not identify PCC over QUIC by network address but rather use connection identifiers to recognize different PCC over QUIC instances.

7. Flow Control

QUIC’s flow control mechanism manages individual streams as well as the entire connection. A QUIC receiver controls the maximum amount of data a sender can transmit on any single stream and across all streams at any given time. PCEPoQ utilizes QUIC’s flow control mechanism, separating control streams from data streams by using a control channel and a data channel.

8. Security Considerations

Security considerations for the QUIC protocol are described in the corresponding section in [RFC9000].

Security considerations for the TLS handshake used to secure QUIC are described in [RFC9001].

9. IANA Considerations

9.1. UDP Port for PCEPoQ

IANA is requested to assign a UDP port (TBD1) from the "Service Name and Transport Protocol Port Number Registry" as follows:

Service Name	PCEPoQ
Port Number	TBD1
Transport Protocol	udp
Description	PCEP over QUIC

Table 2: UDP Port for PCEPoQ

9.2. Registration of the PCEP Identification String

This document creates a new registration for the identification of PCEP in the "TLS Application-Layer Protocol Negotiation (ALPN) Protocol IDs" registry.

The "pcepoq" string identifies PCEP over QUIC:

protocol: PCEP over QUIC

Identification Sequence: "pcepoq"

Specification: This document

9.3. PCEPoQ Capability TLV

IANA is asked to assign a new TLV code from PCEP TLV Type Indicators [RFC5440] for the PCEP over QUIC Capability as follows:

Value	TBD2
Description	PCEPoQ Capability
Reference	This Document
Change Controller	IETF

Table 3: PCEPoQ Capability Registration

Capability:
D: Bit 0, support data channel, this document

10. References

10.1. Normative References

[RFC5440] Vasseur, JP., Ed. and JL. Le Roux, Ed., "Path Computation Element (PCE) Communication Protocol (PCEP)", RFC 5440, DOI 10.17487/RFC5440, March 2009, <https://www.rfc-editor.org/rfc/rfc5440>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/rfc/rfc9000>.

[RFC9001] Thomson, M., Ed. and S. Turner, Ed., "Using TLS to Secure QUIC", RFC 9001, DOI 10.17487/RFC9001, May 2021, <https://www.rfc-editor.org/rfc/rfc9001>.

[RFC9369] Duke, M., "QUIC Version 2", RFC 9369, DOI 10.17487/RFC9369, May 2023, <https://www.rfc-editor.org/rfc/rfc9369>.

[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/rfc/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/rfc/rfc8174>.

10.2. Informative References

[RFC7301] Friedl, S., Popov, A., Langley, A., and E. Stephan, "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension", RFC 7301, DOI 10.17487/RFC7301, July 2014, <https://www.rfc-editor.org/rfc/rfc7301>.

Authors' Addresses

Feng Yang
China Mobile
Xuanwumen West Street
Beijing
China
Email: yangfeng@chinamobile.com

Changwang Lin
New H3C Technologies
Yongjia North Rd
Beijing
China
Email: linchangwang.04414@h3c.com

Samuel Sidor
Cisco Systems, Inc.
Pribinova 10
811 09 Bratislava
Slovakia
Email: ssidor@cisco.com

Tingting Han
China Mobile
Xuanwumen West Street
Beijing
China
Email: hantingting@chinamobile.com