Internet Engineering Task Force (IETF)
Request for Comments: 8899
Updates: 4821, 4960, 6951, 8085, 8261
Category: Standards Track
ISSN: 2070-1721

G. Fairhurst
T. Jones
University of Aberdeen
M. Tüxen
I. Rüngeler
T. Völker
Münster University of Applied Sciences

September 2020

Packetization Layer Path MTU Discovery for Datagram Transports

Abstract

This document specifies Datagram Packetization Layer Path MTU Discovery (DPLPMTUD). This is a robust method for Path MTU Discovery (PMTUD) for datagram Packetization Layers (PLs). It allows a PL, or a datagram application that uses a PL, to discover whether a network path can support the current size of datagram. This can be used to detect and reduce the message size when a sender encounters a packet black hole. It can also probe a network path to discover whether the maximum packet size can be increased. This provides functionality for datagram transports that is equivalent to the PLPMTUD specification for TCP, specified in RFC 4821, which it updates. It also updates the UDP Usage Guidelines to refer to this method for use with UDP datagrams and updates SCTP.

The document provides implementation notes for incorporating Datagram PMTUD into IETF datagram transports or applications that use datagram transports.

This specification updates RFC 4960, RFC 4821, RFC 6951, RFC 8085, and RFC 8261.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc8899.

Copyright Notice

Table of Contents

1. Introduction

The IETF has specified datagram transport using UDP, Stream Control Transmission Protocol (SCTP), and Datagram Congestion Control Protocol (DCCP), as well as protocols layered on top of these transports (e.g., SCTP/UDP, DCCP/UDP, QUIC/UDP) and direct datagram transport over the IP network layer. This document describes a robust method for Path MTU Discovery (PMTUD) that can be used with these transport protocols (or the applications that use their transport service) to discover an appropriate size of packet to use across an Internet path.

1.1. Classical Path MTU Discovery

Classical Path Maximum Transmission Unit Discovery (PMTUD) can be used with any transport that is able to process ICMP Packet Too Big (PTB) messages (e.g., [RFC1191] and [RFC8201]). In this document, the term PTB message is applied to both IPv4 ICMP Unreachable messages (Type 3) that carry the error Fragmentation Needed (Type 3, Code 4) [RFC0792] and ICMPv6 Packet Too Big messages (Type 2) [RFC4443]. When a sender receives a PTB message, it reduces the effective MTU to the value reported as the link MTU in the PTB message. Classical PMTUD specifies a method of periodically increasing the packet size in an attempt to discover an increase in the supported PMTU. The packets sent with a size larger than the current effective PMTU are known as probe packets.

Packets not intended as probe packets are either fragmented to the current effective PMTU, or the attempt to send fails with an error code. Applications can be provided with a primitive to let them read the Maximum Packet Size (MPS), which is derived from the current effective PMTU.

Classical PMTUD is subject to protocol failures. One failure arises when traffic using a packet size larger than the actual PMTU is black-holed (all datagrams larger than the actual PMTU are discarded). This could arise when the PTB messages are not sent back to the sender for some reason (for example, see [RFC2923]).

Examples of where PTB messages are not delivered include the following:

- The generation of ICMP messages is usually rate limited. This could result in no PTB messages being generated to the sender (see Section 2.4 of [RFC4443]).

* ICMP messages can be filtered by middleboxes, including firewalls [RFC4890]. A firewall could be configured with a policy to block incoming ICMP messages, which would prevent reception of PTB messages by a sending endpoint behind this firewall.

* When the router issuing the ICMP message drops a tunneled packet, the resulting ICMP message is directed to the tunnel ingress. This tunnel endpoint is responsible for forwarding the ICMP message, processing the quoted packet within the payload field to remove the effect of the tunnel and returning a correctly formatted ICMP message to the sender [TUNNELS]. Failure to do this prevents the PTB message from reaching the original sender.

* Asymmetry in forwarding can result in there being no return route to the original sender, which would prevent an ICMP message from being delivered to the sender. This issue can also arise when either policy-based or Equal-Cost Multipath (ECMP) routing is used or when a middlebox acts as an application load balancer. An example of which is an ECMP router choosing a path toward the server based on the bytes in the IP payload. In this case, if a packet sent by the server encounters a problem after the ECMP router, then the ECMP router needs to direct any resulting ICMP message toward the original sender.

* There are additional cases where the next-hop destination fails to receive a packet because of its size. This could be due to misconfiguration of the layer 2 path between nodes, for instance the MTU configured in a layer 2 switch, or misconfiguration of the Maximum Receive Unit (MRU). If a packet is dropped by the link, this will not cause a PTB message to be sent to the original sender.

Another failure could result if a node that is not on the network path sends a PTB message that attempts to force a sender to change the effective PMTU [RFC8201]. A sender can protect itself from reacting to such messages by utilizing the quoted packet within a PTB message payload to validate that the received PTB message was generated in response to a packet that had actually originated from the sender. However, there are situations where a sender would be unable to provide this validation. Examples where the validation of the PTB message is not possible include the following:

* When a router issuing the ICMP message implements RFC 792 [RFC0792], it is only required to include the first 64 bits of the IP payload of the packet within the quoted payload. There could be insufficient bytes remaining for the sender to interpret the quoted transport information.

  Note: The recommendation in RFC 1812 [RFC1812] is that IPv4 routers return a quoted packet with as much of the original datagram as possible without the length of the ICMP datagram exceeding 576 bytes. IPv6 routers include as much of the invoking packet as possible without the ICMPv6 packet exceeding 1280 bytes

[RFC4443].

* The use of tunnels and/or encryption can reduce the size of the quoted packet returned to the original source address, increasing the risk that there could be insufficient bytes remaining for the sender to interpret the quoted transport information.

* Even when the PTB message includes sufficient bytes of the quoted packet, the network layer could lack sufficient context to validate the message because validation depends on information about the active transport flows at an endpoint node (e.g., the socket/address pairs being used and other protocol header information).

* When a packet is encapsulated/tunneled over an encrypted transport, the tunnel/encapsulation ingress might have insufficient context, or computational power, to reconstruct the transport header that would be needed to perform validation.

* When an ICMP message is generated by a router in a network segment that has inserted a header into a packet, the quoted packet could contain additional protocol header information that was not included in the original sent packet and that the PL sender does not process or may not know how to process. This could disrupt the ability of the sender to validate this PTB message.

* A Network Address Translation (NAT) device that translates a packet header ought to also translate ICMP messages and update the ICMP-quoted packet [RFC5508] in that message. If this is not correctly translated, then the sender would not be able to associate the message with the PL that originated the packet, and hence this ICMP message cannot be validated.

1.2. Packetization Layer Path MTU Discovery

The term Packetization Layer (PL) has been introduced to describe the layer that is responsible for placing data blocks into the payload of IP packets and selecting an appropriate MPS. This function is often performed by a transport protocol (e.g., DCCP, RTP, SCTP, QUIC) but can also be performed by other encapsulation methods working above the transport layer.

In contrast to PMTUD, Packetization Layer Path MTU Discovery (PLPMTUD) [RFC4821] introduces a method that does not rely upon reception and validation of PTB messages. It is therefore more robust than Classical PMTUD. This has become the recommended approach for implementing discovery of the PMTU [BCP145].

This document updates [RFC4821] to specify the PLPMTUD method for datagram PLs and also updates [BCP145] to refer to the method specified in this document for use with UDP datagrams instead of the method in [RFC4821].

It uses a general strategy in which the PL sends probe packets to search for the largest size of unfragmented datagram that can be sent over a network path. Probe packets are sent to explore using a larger packet size. If a probe packet is successfully delivered (as determined by the PL), then the PLPMTU is raised to the size of the successful probe. If a black hole is detected (e.g., where packets of size PLPMTU are consistently not received), the method reduces the PLPMTU.

Datagram PLPMTUD introduces flexibility in implementation. At one extreme, it can be configured to only perform black hole detection and recovery with increased robustness compared to Classical PMTUD. At the other extreme, all PTB processing can be disabled, and PLPMTUD replaces Classical PMTUD.

PLPMTUD can also include additional consistency checks without increasing the risk that data is lost when probing to discover the Path MTU. For example, information available at the PL, or higher layers, enables received PTB messages to be validated before being utilized.

1.3. Path MTU Discovery for Datagram Services

Section 5 of this document presents a set of algorithms for datagram protocols to discover the largest size of unfragmented datagram that can be sent over a network path. The method relies upon features of the PL described in Section 3 and applies to transport protocols operating over IPv4 and IPv6. It does not require cooperation from the lower layers, although it can utilize PTB messages when these received messages are made available to the PL.

The message size guidelines in Section 3.2 of the UDP Usage Guidelines [BCP145] state that "an application SHOULD either use the Path MTU information provided by the IP layer or implement Path MTU Discovery (PMTUD)" but do not provide a mechanism for discovering the largest size of unfragmented datagram that can be used on a network path. The present document updates RFC 8085 to specify this method in place of PLPMTUD [RFC4821] and provides a mechanism for sharing the discovered largest size as the MPS (see Section 4.4).

Section 10.2 of [RFC4821] recommended a PLPMTUD probing method for the Stream Control Transport Protocol (SCTP). SCTP utilizes probe packets consisting of a minimal-sized HEARTBEAT chunk bundled with a PAD chunk as defined in [RFC4820]. However, RFC 4821 did not provide a complete specification. The present document replaces that description by providing a complete specification.

The Datagram Congestion Control Protocol (DCCP) [RFC4340] requires implementations to support Classical PMTUD and states that a DCCP sender "MUST maintain the MPS allowed for each active DCCP session". It also defines the current congestion control MPS (CCMPS) supported by a network path. This recommends use of PMTUD and suggests use of control packets (DCCP-Sync) as path probe packets because they do not risk application data loss. The method defined in this specification can be used with DCCP.

Section 4 and Section 5 define the protocol mechanisms and specification for Datagram Packetization Layer Path MTU Discovery (DPLPMTUD).

Section 6 specifies the method for datagram transports and provides information to enable the implementation of PLPMTUD with other datagram transports and applications that use datagram transports.

Section 6 also provides recommendations for SCTP endpoints, updating [RFC4960], [RFC6951], and [RFC8261] to use the method specified in this document instead of the method in [RFC4821].

2. Terminology

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.

The following terminology is defined. Relevant terms are directly copied from [RFC4821], and the definitions in [RFC1122] apply.

Acknowledged PL: A PL that includes a mechanism that can confirm successful delivery of datagrams to the remote PL endpoint (e.g., SCTP). Typically, the PL receiver returns acknowledgments corresponding to the received datagrams, which can be utilized to detect black-holing of packets (c.f., Unacknowledged PL).

Actual PMTU: The actual PMTU is the PMTU of a network path between a sender PL and a destination PL, which the DPLPMTUD algorithm seeks to determine.

Black Hole: A black hole is encountered when a sender is unaware that packets are not being delivered to the destination endpoint. Two types of black hole are relevant to DPLPMTUD:

- Packets encounter a packet black hole when packets are not delivered to the destination endpoint (e.g., when the sender transmits packets of a particular size with a previously known effective PMTU, and they are discarded by the network).

- An ICMP black hole is encountered when the sender is unaware that packets are not delivered to the destination endpoint because PTB messages are not received by the originating PL sender.

Classical Path MTU Discovery: Classical PMTUD is a process described in [RFC1191] and [RFC8201] in which nodes rely on PTB messages to learn the largest size of unfragmented packet that can be used across a network path.

Datagram: A datagram is a transport-layer protocol data unit, transmitted in the payload of an IP packet.

DPLPMTUD: Datagram Packetization Layer Path MTU Discovery (DPLPMTUD), PLPMTUD performed using a datagram transport protocol.

Effective PMTU: The effective PMTU is the current estimated value for PMTU that is used by a PMTUD. This is equivalent to the PLPMTU derived by PLPMTUD plus the size of any headers added below the PL, including the IP layer headers.

EMTU_S: The effective MTU for sending (EMTU_S) is defined in [RFC1122] as "the maximum IP datagram size that may be sent, for a particular combination of IP source and destination addresses...".

EMTU_R: The effective MTU for receiving (EMTU_R) is designated in [RFC1122] as "the largest datagram size that can be reassembled".

Link: A link is a communication facility or medium over which nodes can communicate at the link layer, i.e., a layer below the IP layer. Examples are Ethernet LANs and Internet (or higher) layer tunnels.

Link MTU: The link Maximum Transmission Unit (MTU) is the size in bytes of the largest IP packet, including the IP header and payload, that can be transmitted over a link. Note that this could more properly be called the IP MTU, to be consistent with how other standards organizations use the acronym. This includes the IP header but excludes link layer headers and other framing that is not part of IP or the IP payload. Other standards organizations generally define the link MTU to include the link layer headers. This specification continues the requirement in [RFC4821] that states, "All links MUST enforce their MTU: links that might non-deterministically deliver packets that are larger than their rated MTU MUST consistently discard such packets."

MAX_PLPMTU: The MAX_PLPMTU is the largest size of PLPMTU that DPLPMTUD will attempt to use (see the constants defined in Section 5.1.2).

MIN_PLPMTU: The MIN_PLPMTU is the smallest size of PLPMTU that DPLPMTUD will attempt to use (see the constants defined in Section 5.1.2).

MPS: The Maximum Packet Size (MPS) is the largest size of application data block that can be sent across a network path by a PL using a single datagram (see Section 4.4).

MSL: The Maximum Segment Lifetime (MSL) is the maximum delay a packet is expected to experience across a path, taken as 2 minutes [BCP145].

Packet: A packet is the IP header(s) and any extension headers/options plus the IP payload.

Packetization Layer (PL): The PL is a layer of the network stack that places data into packets and performs transport protocol functions. Examples of a PL include TCP, SCTP, SCTP over UDP, SCTP over DTLS, or QUIC.

Path: The path is the set of links and routers traversed by a packet between a source node and a destination node by a particular flow.

Path MTU (PMTU): The Path MTU (PMTU) is the minimum of the link MTU of all the links forming a network path between a source node and a destination node, as used by PMTUD.

PTB: In this document, the term PTB message is applied to both IPv4 ICMP Unreachable messages (Type 3) that carry the error Fragmentation Needed (Type 3, Code 4) [RFC0792] and ICMPv6 Packet Too Big messages (Type 2) [RFC4443].

PTB_SIZE: The PTB_SIZE is a value reported in a validated PTB message that indicates next-hop link MTU of a router along the path.

PL_PTB_SIZE: The size reported in a validated PTB message, reduced by the size of all headers added by layers below the PL.

PLPMTU: The Packetization Layer PMTU is an estimate of the largest size of PL datagram that can be sent by a path, controlled by PLPMTUD.

PLPMTUD: Packetization Layer Path MTU Discovery (PLPMTUD), the method described in this document for datagram PLs, which is an extension to Classical PMTU Discovery.

Probe packet: A probe packet is a datagram sent with a purposely chosen size (typically the current PLPMTU or larger) to detect if packets of this size can be successfully sent end-to-end across the network path.

Unacknowledged PL: A PL that does not itself provide a mechanism to confirm delivery of datagrams to the remote PL endpoint (e.g., UDP), and therefore requires DPLPMTUD to provide a mechanism to detect black-holing of packets (c.f., Acknowledged PL).

3. Features Required to Provide Datagram PLPMTUD

The principles expressed in [RFC4821] apply to the use of the technique with any PL. TCP PLPMTUD has been defined using standard TCP protocol mechanisms. Unlike TCP, a datagram PL requires additional mechanisms and considerations to implement PLPMTUD.

The requirements for datagram PLPMTUD are:

1. Managing the PLPMTU: For datagram PLs, the PLPMTU is managed by DPLPMTUD. A PL MUST NOT send a datagram (other than a probe packet) with a size at the PL that is larger than the current PLPMTU.

2. Probe packets: The network interface below the PL is REQUIRED to provide a way to transmit a probe packet that is larger than the PLPMTU. In IPv4, a probe packet MUST be sent with the Don't Fragment (DF) bit set in the IP header and without network layer endpoint fragmentation. In IPv6, a probe packet is always sent without source fragmentation (as specified in Section 5.4 of [RFC8201]).

3. Reception feedback: The destination PL endpoint is REQUIRED to provide a feedback method that indicates to the DPLPMTUD sender when a probe packet has been received by the destination PL endpoint. Section 6 provides examples of how a PL can provide this acknowledgment of received probe packets.

4. Probe loss recovery: It is RECOMMENDED to use probe packets that do not carry any user data that would require retransmission if lost. Most datagram transports permit this. If a probe packet contains user data requiring retransmission in case of loss, the PL (or layers above) is REQUIRED to arrange any retransmission and/or repair of any resulting loss. The PL is REQUIRED to be robust in the case where probe packets are lost due to other reasons (including link transmission error, congestion).

5. PMTU parameters: A DPLPMTUD sender is RECOMMENDED to utilize information about the maximum size of packet that can be transmitted by the sender on the local link (e.g., the local link MTU). A PL sender MAY utilize similar information about the maximum size of network-layer packet that a receiver can accept when this is supplied (note this could be less than EMTU_R). This avoids implementations trying to send probe packets that cannot be transferred by the local link. Too high of a value could reduce the efficiency of the search algorithm. Some applications also have a maximum transport protocol data unit (PDU) size, in which case there is no benefit from probing for a size larger than this (unless a transport allows multiplexing multiple applications' PDUs into the same datagram).

6. Processing PTB messages: A DPLPMTUD sender MAY optionally utilize PTB messages received from the network layer to help identify when a network path does not support the current size of probe packet. Any received PTB message MUST be validated before it is used to update the PLPMTU discovery information [RFC8201]. This validation confirms that the PTB message was sent in response to a packet originated by the sender and needs to be performed before the PLPMTU discovery method reacts to the PTB message. A PTB message MUST NOT be used to increase the PLPMTU [RFC8201] but could trigger a probe to test for a larger PLPMTU. A valid PTB_SIZE is converted to a PL_PTB_SIZE before it is to be used in the DPLPMTUD state machine. A PL_PTB_SIZE that is greater than that currently probed SHOULD be ignored. (This PTB message ought to be discarded without further processing but could be utilized as an input that enables a resilience mode).

7. Probing and congestion control: A PL MAY use a congestion controller to decide when to send a probe packet. If transmission of probe packets is limited by the congestion controller, this could result in transmission of probe packets being delayed or suspended during congestion. When the transmission of probe packets is not controlled by the congestion controller, the interval between probe packets MUST be at least one RTT. Loss of a probe packet SHOULD NOT be treated as an indication of congestion and SHOULD NOT trigger a congestion control reaction [RFC4821] because this could result in unnecessary reduction of the sending rate. An update to the

PLPMTU (or MPS) MUST NOT increase the congestion window measured in bytes [RFC4821]. Therefore, an increase in the packet size does not cause an increase in the data rate in bytes per second. A PL that maintains the congestion window in terms of a limit to the number of outstanding fixed-size packets SHOULD adapt this limit to compensate for the size of the actual packets. The transmission of probe packets can interact with the operation of a PL that performs burst mitigation or pacing, and the PL could need transmission of probe packets to be regulated by these methods.

8. Probing and flow control: Flow control at the PL concerns the end-to-end flow of data using the PL service. Flow control SHOULD NOT apply to DPLPMTU when probe packets use a design that does not carry user data to the remote application.

9. Shared PLPMTU state: The PMTU value calculated from the PLPMTU MAY also be stored with the corresponding entry associated with the destination in the IP layer cache and used by other PL instances. The specification of PLPMTUD [RFC4821] states, "If PLPMTUD updates the MTU for a particular path, all Packetization Layer sessions that share the path representation (as described in Section 5.2) SHOULD be notified to make use of the new MTU". Such methods MUST be robust to the wide variety of underlying network forwarding behaviors. Section 5.2 of [RFC8201] provides guidance on the caching of PMTU information and also the relation to IPv6 flow labels.

In addition, the following principles are stated for design of a DPLPMTUD method:

* A PL MAY be designed to segment data blocks larger than the MPS into multiple datagrams. However, not all datagram PLs support segmentation of data blocks. It is RECOMMENDED that methods avoid forcing an application to use an arbitrary small MPS for transmission while the method is searching for the currently supported PLPMTU. A reduced MPS can adversely impact the performance of an application.

* To assist applications in choosing a suitable data block size, the PL is RECOMMENDED to provide a primitive that returns the MPS derived from the PLPMTU to the higher layer using the PL. The value of the MPS can change following a change in the path or loss of probe packets.

* Path validation: It is RECOMMENDED that methods are robust to path changes that could have occurred since the path characteristics were last confirmed and to the possibility of inconsistent path information being received.

* Datagram reordering: A method is REQUIRED to be robust to the possibility that a flow encounters reordering or that the traffic (including probe packets) is divided over more than one network path.

* Datagram delay and duplication: The feedback mechanism is REQUIRED

to be robust to the possibility that packets could be significantly delayed or duplicated along a network path.

- When to probe: It is RECOMMENDED that methods determine whether the path has changed since it last measured the path. This can help determine when to probe the path again.

4. DPLPMTUD Mechanisms

This section lists the protocol mechanisms used in this specification.

4.1. PLPMTU Probe Packets

The DPLPMTUD method relies upon the PL sender being able to generate probe packets with a specific size. TCP is able to generate these probe packets by choosing to appropriately segment data being sent [RFC4821]. In contrast, a datagram PL that constructs a probe packet has to either request an application to send a data block that is larger than that generated by an application, or to utilize padding functions to extend a datagram beyond the size of the application data block. Protocols that permit exchange of control messages (without an application data block) can generate a probe packet by extending a control message with padding data. The total size of a probe packet includes all headers and padding added to the payload data being sent (e.g., including protocol option fields, security-related fields such as an Authenticated Encryption with Associated Data (AEAD) tag, and TLS record layer padding).

A receiver is REQUIRED to be able to distinguish an in-band data block from any added padding. This is needed to ensure that any added padding is not passed on to an application at the receiver.

This results in three possible ways that a sender can create a probe packet:

Probing using padding data: A probe packet that contains only control information together with any padding, which is needed to inflate to the size of the probe packet. Since these probe packets do not carry an application-supplied data block, they do not typically require retransmission, although they do still consume network capacity and incur endpoint processing.

Probing using application data and padding data: A probe packet that contains a data block supplied by an application that is combined with padding to inflate the length of the datagram to the size of the probe packet.

Probing using application data: A probe packet that contains a data block supplied by an application that matches the size of the probe packet. This method requests the application to issue a data block of the desired probe size.

A PL that uses a probe packet carrying application data and that needs protection from the loss of this probe packet could perform transport-layer retransmission/repair of the data block (e.g., by retransmitting after loss is detected or by duplicating the data block in a datagram without the padding data). This retransmitted data block might possibly need to be sent using a smaller PLPMTU, which could force the PL to use a smaller packet size to traverse the end-to-end path. (This could utilize endpoint network-layer fragmentation or a PL that can resegment the data block into multiple datagrams).

DPLPMTUD MAY choose to use only one of these methods to simplify the implementation.

Probe messages sent by a PL MUST contain enough information to uniquely identify the probe within the Maximum Segment Lifetime (e.g., including a unique identifier from the PL or the DPLPMTUD implementation), while being robust to reordering and replay of probe response and PTB messages.

4.2. Confirmation of Probed Packet Size

The PL needs a method to determine (confirm) when probe packets have been successfully received end-to-end across a network path.

Transport protocols can include end-to-end methods that detect and report reception of specific datagrams that they send (e.g., DCCP, SCTP, and QUIC provide keep-alive/heartbeat features). When supported, this mechanism MAY also be used by DPLPMTUD to acknowledge reception of a probe packet.

A PL that does not acknowledge data reception (e.g., UDP and UDP-Lite) is unable itself to detect when the packets that it sends are discarded because their size is greater than the actual PMTU. These PLs need to rely on an application protocol to detect this loss.

Section 6 specifies this function for a set of IETF-specified protocols.

4.3. Black Hole Detection and Reducing the PLPMTU

The description that follows uses the set of constants defined in Section 5.1.2 and variables defined in Section 5.1.3.

Black hole detection is triggered by an indication that the network path could be unable to support the current PLPMTU size.

There are three indicators that can be used to detect black holes:

* A validated PTB message can be received that indicates a PL_PTB_SIZE less than the current PLPMTU. A DPLPMTUD method MUST NOT rely solely on this method.

* A PL can use the DPLPMTUD probing mechanism to periodically generate probe packets of the size of the current PLPMTU (e.g., using the CONFIRMATION_TIMER, Section 5.1.1). A timer tracks whether acknowledgments are received. Successive loss of probes is an indication that the current path no longer supports the PLPMTU (e.g., when the number of probe packets sent without

receiving an acknowledgment, PROBE_COUNT, becomes greater than MAX_PROBES).

- A PL can utilize an event that indicates the network path no longer sustains the sender's PLPMTU size. This could use a mechanism implemented within the PL to detect excessive loss of data sent with a specific packet size and then conclude that this excessive loss could be a result of an invalid PLPMTU (as in PLPMTUD for TCP [RFC4821]).

The three methods can result in different transmission patterns for packet probes and are expected to result in different responsiveness following a change in the actual PMTU.

A PL MAY inhibit sending probe packets when no application data has been sent since the previous probe packet. A PL that resumes sending user data MAY continue PLPMTU discovery for each path. This allows it to use an up-to-date PLPMTU. However, this could result in additional packets being sent.

When the method detects that the current PLPMTU is not supported, DPLPMTUD sets a lower PLPMTU and a lower MPS. The PL then confirms that the new PLPMTU can be successfully used across the path. A probe packet could need to be smaller than the size of the data block generated by the application.

4.4. The Maximum Packet Size (MPS)

The result of probing determines a usable PLPMTU, which is used to set the MPS used by the application. The MPS is smaller than the PLPMTU because it is reduced by the size of PL headers (including the overhead of security-related fields such as an AEAD tag and TLS record layer padding). The relationship between the MPS and the PLPMTUD is illustrated in Figure 1.

```
Any additional
  headers         .--- MPS -----.
         |        |             |
         v        v             v
  +------------------------------+
  | IP | ** | PL | protocol data |
  +------------------------------+

             <----- PLPMTU ----->
  <---------- PMTU -------------->
```

Figure 1: Relationship between MPS and PLPMTU

A PL is unable to send a packet (other than a probe packet) with a size larger than the current PLPMTU at the network layer. To avoid this, a PL MAY be designed to segment data blocks larger than the MPS into multiple datagrams.

DPLPMTUD seeks to avoid IP fragmentation. An attempt to send a data block larger than the MPS will therefore fail if a PL is unable to segment data. To determine the largest data block that can be sent, a PL SHOULD provide applications with a primitive that returns the MPS, derived from the current PLPMTU.

If DPLPMTUD results in a change to the MPS, the application needs to adapt to the new MPS. A particular case can arise when packets have been sent with a size less than the MPS and the PLPMTU was subsequently reduced. If these packets are lost, the PL MAY segment the data using the new MPS. If a PL is unable to resegment a previously sent datagram (e.g., [RFC4960]), then the sender either discards the datagram or could perform retransmission using network-layer fragmentation to form multiple IP packets not larger than the PLPMTU. For IPv4, the use of endpoint fragmentation by the sender is preferred over clearing the DF bit in the IPv4 header. Operational experience reveals that IP fragmentation can reduce the reliability of Internet communication [RFC8900], which may reduce the probability of successful retransmission.

4.5. Disabling the Effect of PMTUD

A PL implementing this specification MUST suspend network layer processing of outgoing packets that enforces a PMTU [RFC1191][RFC8201] for each flow utilizing DPLPMTUD and instead use DPLPMTUD to control the size of packets that are sent by a flow. This removes the need for the network layer to drop or to fragment sent packets that have a size greater than the PMTU.

4.6. Response to PTB Messages

This method requires the DPLPMTUD sender to validate any received PTB message before using the PTB information. The response to a PTB message depends on the PL_PTB_SIZE calculated from the PTB_SIZE in the PTB message, the state of the PLPMTUD state machine, and the IP protocol being used.

Section 4.6.1 describes validation for both IPv4 ICMP Unreachable messages (Type 3) and ICMPv6 Packet Too Big messages, both of which are referred to as PTB messages in this document.

4.6.1. Validation of PTB Messages

This section specifies utilization and validation of PTB messages.

- A simple implementation MAY ignore received PTB messages, and in this case, the PLPMTU is not updated when a PTB message is received.

- A PL that supports PTB messages MUST validate these messages before they are further processed.

A PL that receives a PTB message from a router or middlebox performs ICMP validation (see Section 4 of [RFC8201] and Section 5.2 of [BCP145]). Because DPLPMTUD operates at the PL, the PL needs to check that each received PTB message is received in response to a packet transmitted by the endpoint PL performing DPLPMTUD.

The PL MUST check the protocol information in the quoted packet carried in an ICMP PTB message payload to validate the message originated from the sending node. This validation includes determining that the combination of the IP addresses, the protocol, the source port, and destination port match those returned in the quoted packet -- this is also necessary for the PTB message to be passed to the corresponding PL.

The validation SHOULD utilize information that is not simple for an off-path attacker to determine [BCP145]. For example, it could check the value of a protocol header field known only to the two PL endpoints. A datagram application that uses well-known source and destination ports ought to also rely on other information to complete this validation.

These checks are intended to provide protection from packets that originate from a node that is not on the network path. A PTB message that does not complete the validation MUST NOT be further utilized by the DPLPMTUD method, as discussed in the Security Considerations section (Section 8).

Section 4.6.2 describes this processing of PTB messages.

4.6.2. Use of PTB Messages

PTB messages that have been validated MAY be utilized by the DPLPMTUD algorithm but MUST NOT be used directly to set the PLPMTU.

Before using the size reported in the PTB message, it must first be converted to a PL_PTB_SIZE. The PL_PTB_SIZE is smaller than the PTB_SIZE because it is reduced by headers below the PL, including any IP options or extensions added to the PL packet.

A method that utilizes these PTB messages can improve the speed at which the algorithm detects an appropriate PLPMTU by triggering an immediate probe for the PL_PTB_SIZE (resulting in a network-layer packet of size PTB_SIZE), compared to one that relies solely on probing using a timer-based search algorithm.

A set of checks are intended to provide protection from a router that reports an unexpected PTB_SIZE. The PL also needs to check that the indicated PL_PTB_SIZE is less than the size used by probe packets and at least the minimum size accepted.

This section provides a summary of how PTB messages can be utilized, using the set of constants defined in Section 5.1.2. This processing depends on the PL_PTB_SIZE and the current value of a set of variables:

PL_PTB_SIZE < MIN_PLPMTU

- Invalid PL_PTB_SIZE, see Section 4.6.1.
- PTB message ought to be discarded without further processing (i.e., PLPMTU is not modified).
- The information could be utilized as an input that triggers the enabling of a resilience mode (see Section 5.3.3).

MIN_PLPMTU < PL_PTB_SIZE < BASE_PLPMTU

- A robust PL MAY enter an error state (see Section 5.2) for an IPv4 path when the PL_PTB_SIZE reported in the PTB message is larger than or equal to 68 bytes [RFC0791] and when this is less than the BASE_PLPMTU.

- A robust PL MAY enter an error state (see Section 5.2) for an IPv6 path when the PL_PTB_SIZE reported in the PTB message is larger than or equal to 1280 bytes [RFC8200] and when this is less than the BASE_PLPMTU.

BASE_PLPMTU <= PL_PTB_SIZE < PLPMTU

- This could be an indication of a black hole. The PLPMTU SHOULD be set to BASE_PLPMTU (the PLPMTU is reduced to the BASE_PLPMTU to avoid unnecessary packet loss when a black hole is encountered).

- The PL ought to start a search to quickly discover the new PLPMTU. The PL_PTB_SIZE reported in the PTB message can be used to initialize a search algorithm.

PLPMTU < PL_PTB_SIZE < PROBED_SIZE

- The PLPMTU continues to be valid, but the size of a packet used to search (PROBED_SIZE) was larger than the actual PMTU.

- The PLPMTU is not updated.

- The PL can use the reported PL_PTB_SIZE from the PTB message as the next search point when it resumes the search algorithm.

PL_PTB_SIZE >= PROBED_SIZE

- Inconsistent network signal.

- PTB message ought to be discarded without further processing (i.e., PLPMTU is not modified).

- The information could be utilized as an input to trigger the enabling of a resilience mode.

5. Datagram Packetization Layer PMTUD

This section specifies Datagram PLPMTUD (DPLPMTUD). The method can be introduced at various points (as indicated with * in Figure 2) in the IP protocol stack to discover the PLPMTU so that an application can utilize an appropriate MPS for the current network path.

DPLPMTUD SHOULD only be performed at one layer between a pair of endpoints. Therefore, an upper PL or application should avoid using DPLPMTUD when this is already enabled in a lower layer. A PL MUST adjust the MPS indicated by DPLPMTUD to account for any additional overhead introduced by the PL.

```
+----------------------+
|     Application*     |
+-----+------------+---+
         |            |
     +---+--+      +--+--+
     | QUIC*|      |SCTP*|
     +---+--+      +-+-+-+
         |           | |
         +---+  +-----+ |
             |  |      |
           +-+--+-+    |
           | UDP  |    |
           +---+--+    |
               |       |
   +-----------+-------+--+
   |  Network Interface   |
   +----------------------+
```

Figure 2: Examples Where DPLPMTUD Can Be Implemented

The central idea of DPLPMTUD is probing by a sender. Probe packets are sent to find the maximum size of user message that can be completely transferred across the network path from the sender to the destination.

The following sections identify the components needed for implementation, provide an overview of the phases of operation, and specify the state machine and search algorithm.

5.1. DPLPMTUD Components

This section describes the timers, constants, and variables of DPLPMTUD.

5.1.1. Timers

The method utilizes up to three timers:

PROBE_TIMER: The PROBE_TIMER is configured to expire after a period longer than the maximum time to receive an acknowledgment to a probe packet. This value MUST NOT be smaller than 1 second and SHOULD be larger than 15 seconds. Guidance on the selection of the timer value is provided in Section 3.1.1 of the UDP Usage Guidelines [BCP145].

PMTU_RAISE_TIMER: The PMTU_RAISE_TIMER is configured to the period a sender will continue to use the current PLPMTU, after which it reenters the Search Phase. This timer has a period of 600 seconds, as recommended by PLPMTUD [RFC4821].

DPLPMTUD MAY inhibit sending probe packets when no application data has been sent since the previous probe packet. A PL preferring to use an up-to-date PMTU once user data is sent again can choose to continue PMTU discovery for each path. However, this will result in sending additional packets.

CONFIRMATION_TIMER: When an acknowledged PL is used, this timer MUST NOT be used. For other PLs, the CONFIRMATION_TIMER is configured to the period a PL sender waits before confirming the current

PLPMTU is still supported. This is less than the PMTU_RAISE_TIMER and used to decrease the PLPMTU (e.g., when a black hole is encountered). Confirmation needs to be frequent enough when data is flowing that the sending PL does not black hole extensive amounts of traffic. Guidance on selection of the timer value are provided in Section 3.1.1 of the UDP Usage Guidelines [BCP145].

DPLPMTUD MAY inhibit sending probe packets when no application data has been sent since the previous probe packet. A PL preferring to use an up-to-date PMTU once user data is sent again, can choose to continue PMTU discovery for each path. However, this could result in sending additional packets.

DPLPMTUD specifies various timers; however, an implementation could choose to realize these timer functions using a single timer.

5.1.2. Constants

The following constants are defined:

MAX_PROBES: The MAX_PROBES is the maximum value of the PROBE_COUNT counter (see Section 5.1.3). MAX_PROBES represents the limit for the number of consecutive probe attempts of any size. Search algorithms benefit from a MAX_PROBES value greater than 1 because this can provide robustness to isolated packet loss. The default value of MAX_PROBES is 3.

MIN_PLPMTU: The MIN_PLPMTU is the smallest size of PLPMTU that DPLPMTUD will attempt to use. An endpoint could need to configure the MIN_PLPMTU to provide space for extension headers and other encapsulations at layers below the PL. This value can be interface and path dependent. For IPv6, this size is greater than or equal to the size at the PL that results in an 1280-byte IPv6 packet, as specified in [RFC8200]. For IPv4, this size is greater than or equal to the size at the PL that results in an 68-byte IPv4 packet. Note: An IPv4 router is required to be able to forward a datagram of 68 bytes without further fragmentation. This is the combined size of an IPv4 header and the minimum fragment size of 8 bytes. In addition, receivers are required to be able to reassemble fragmented datagrams at least up to 576 bytes, as stated in Section 3.3.3 of [RFC1122].

MAX_PLPMTU: The MAX_PLPMTU is the largest size of PLPMTU. This has to be less than or equal to the maximum size of the PL packet that can be sent on the outgoing interface (constrained by the local interface MTU). When known, this also ought to be less than the maximum size of PL packet that can be received by the remote endpoint (constrained by EMTU_R). It can be limited by the design or configuration of the PL being used. An application, or PL, MAY choose a smaller MAX_PLPMTU when there is no need to send packets larger than a specific size.

BASE_PLPMTU: The BASE_PLPMTU is a configured size expected to work for most paths. The size is equal to or larger than the MIN_PLPMTU and smaller than the MAX_PLPMTU. For most PLs, a suitable BASE_PLPMTU will be larger than 1200 bytes. When using

IPv4, there is no currently equivalent size specified, and a default BASE_PLPMTU of 1200 bytes is RECOMMENDED.

5.1.3. Variables

This method utilizes a set of variables:

PROBED_SIZE: The PROBED_SIZE is the size of the current probe packet as determined at the PL. This is a tentative value for the PLPMTU, which is awaiting confirmation by an acknowledgment.

PROBE_COUNT: The PROBE_COUNT is a count of the number of successive unsuccessful probe packets that have been sent. Each time a probe packet is acknowledged, the value is set to zero. (Some probe loss is expected while searching, therefore loss of a single probe is not an indication of a PMTU problem.)

Figure 3 illustrates the relationship between the packet size constants and variables at a point of time when the DPLPMTUD algorithm performs path probing to increase the size of the PLPMTU. A probe packet has been sent of size PROBED_SIZE. Once this is acknowledged, the PLPMTU will raise to PROBED_SIZE, allowing the DPLPMTUD algorithm to further increase PROBED_SIZE toward sending a probe with the size of the actual PMTU.

```
        MIN_PLPMTU                                MAX_PLPMTU
          <-------------------------------------------->
                         |        |     |
                         v        |     |
                   BASE_PLPMTU    |     v
                                  |  PROBED_SIZE
                                  v
                                PLPMTU
```

Figure 3: Relationships between Packet Size Constants and Variables

5.1.4. Overview of DPLPMTUD Phases

This section provides a high-level, informative view of the DPLPMTUD method, by describing the movement of the method through several phases of operation. More detail is available in the state machine, Section 5.2.

```
                       +------+
              +------->| Base |-----------------+ Connectivity
              |        +------+                 | or BASE_PLPMTU
              |           |                     | confirmation failed
              |           |                     v
              |           | Connectivity    +-------+
              |           | and BASE_PLPMTU | Error |
              |           | confirmed       +-------+
              |           |                     | Consistent
              |           v                     | connectivity
   Black Hole |       +--------+                | and BASE_PLPMTU
    detected  |       | Search |<---------------+ confirmed
              |       +--------+
           |          ^  |
           |          |  |
           |    Raise |  | Search
           |    timer |  | algorithm
           |  expired |  | completed
           |          |  |
           |          |  v
           |   +-----------------+
           +---| Search Complete |
               +-----------------+
```

Figure 4: DPLPMTUD Phases

Base: The Base Phase confirms connectivity to the remote peer using packets of the BASE_PLPMTU. The confirmation of connectivity is implicit for a connection-oriented PL (where it can be performed in a PL connection handshake). A connectionless PL sends a probe packet and uses acknowledgment of this probe packet to confirm that the remote peer is reachable.

The sender also confirms that BASE_PLPMTU is supported across the network path. This may be achieved by using a PL mechanism (e.g., using a handshake packet of size BASE_PLPMTU) or by sending a probe packet of size BASE_PLPMTU and confirming that this is received.

A probe packet of size BASE_PLPMTU can be sent immediately on the initial entry to the Base Phase (following a connectivity check). A PL that does not wish to support a path with a PLPMTU less than BASE_PLPMTU can simplify the phase into a single step by performing the connectivity checks with a probe of the BASE_PLPMTU size.

Once confirmed, DPLPMTUD enters the Search Phase. If the Base Phase fails to confirm the BASE_PLPMTU, DPLPMTUD enters the Error Phase.

Search: The Search Phase utilizes a search algorithm to send probe packets to seek to increase the PLPMTU. The algorithm concludes when it has found a suitable PLPMTU by entering the Search Complete Phase.

A PL could respond to PTB messages using the PTB to advance or terminate the search, see Section 4.6.

Search Complete: The Search Complete Phase is entered when the PLPMTU is supported across the network path. A PL can use a CONFIRMATION_TIMER to periodically repeat a probe packet for the current PLPMTU size. If the sender is unable to confirm reachability (e.g., if the CONFIRMATION_TIMER expires) or the PL signals a lack of reachability, a black hole has been detected and DPLPMTUD enters the Base Phase.

The PMTU_RAISE_TIMER is used to periodically resume the Search Phase to discover if the PLPMTU can be raised. Black hole detection causes the sender to enter the Base Phase.

Error: The Error Phase is entered when there is conflicting or invalid PLPMTU information for the path (e.g., a failure to support the BASE_PLPMTU) that causes DPLPMTUD to be unable to progress, and the PLPMTU is lowered.

DPLPMTUD remains in the Error Phase until a consistent view of the path can be discovered and it has also been confirmed that the path supports the BASE_PLPMTU (or DPLPMTUD is suspended).

A method that only reduces the PLPMTU to a suitable size would be sufficient to ensure reliable operation but can be very inefficient when the actual PMTU changes or when the method (for whatever reason) makes a suboptimal choice for the PLPMTU.

A full implementation of DPLPMTUD provides an algorithm enabling the DPLPMTUD sender to increase the PLPMTU following a change in the characteristics of the path, such as when a link is reconfigured with a larger MTU, or when there is a change in the set of links traversed by an end-to-end flow (e.g., after a routing or path failover decision).

5.2. State Machine

A state machine for DPLPMTUD is depicted in Figure 5. If multipath or multihoming is supported, a state machine is needed for each path.

Note: Not all changes are shown to simplify the diagram.

```
      | Start   | PL indicates loss
      |         |  of connectivity
      v         v
   +---------------+                                   +---------------+
   |    DISABLED   |                                   |     ERROR     |
   +---------------+               PROBE_TIMER expiry: +---------------+
           | PL indicates     PROBE_COUNT = MAX_PROBES or    ^      |
           | connectivity  PTB: PL_PTB_SIZE < BASE_PLPMTU    |      |
           +--------------------+         +------------------+      |
                                |         |                         |
                                v         |       BASE_PLPMTU Probe |
                             +---------------+          acked       |
                             |      BASE     |--------------------->+
                             +---------------+                      |
                                ^ |    ^  ^                         |
            Black hole detected | |    |  | Black hole detected     |
           +--------------------+ |    |  +--------------------+    |
           |                      +----+                       |    |
           |                PROBE_TIMER expiry:                |    |
           |             PROBE_COUNT < MAX_PROBES              |    |
           |                                                   |    |
           |               PMTU_RAISE_TIMER expiry             |    |
           |    +-----------------------------------------+    |    |
           |    |                                         |    |    |
           |    |                                         v    |    v
   +---------------+                                   +---------------+
|SEARCH_COMPLETE|                                   |   SEARCHING   |
+---------------+                                   +---------------+
   |    ^    ^                                         |    |    ^
   |    |    |                                         |    |    |
   |    |    +-----------------------------------------+    |    |
   |    |            MAX_PLPMTU Probe acked or              |    |
   |    |  PROBE_TIMER expiry: PROBE_COUNT = MAX_PROBES or  |    |
   +----+            PTB: PL_PTB_SIZE = PLPMTU              +----+
CONFIRMATION_TIMER expiry:                        PROBE_TIMER expiry:
PROBE_COUNT < MAX_PROBES or               PROBE_COUNT < MAX_PROBES or
     PLPMTU Probe acked                           Probe acked or PTB:
                                   PLPMTU < PL_PTB_SIZE < PROBED_SIZE
```

Figure 5: State Machine for Datagram PLPMTUD

The following states are defined:

DISABLED: The DISABLED state is the initial state before probing has started. It is also entered from any other state, when the PL indicates loss of connectivity. This state is left once the PL indicates connectivity to the remote PL. When transitioning to the BASE state, a probe packet of size BASE_PLPMTU can be sent immediately.

BASE: The BASE state is used to confirm that the BASE_PLPMTU size is supported by the network path and is designed to allow an application to continue working when there are transient reductions in the actual PMTU. It also seeks to avoid long periods when a sender searching for a larger PLPMTU is unaware that packets are not being delivered due to a packet or ICMP black hole.

On entry, the PROBED_SIZE is set to the BASE_PLPMTU size, and the PROBE_COUNT is set to zero.

Each time a probe packet is sent, the PROBE_TIMER is started. The state is exited when the probe packet is acknowledged, and the PL sender enters the SEARCHING state.

The state is also left when the PROBE_COUNT reaches MAX_PROBES or a received PTB message is validated. This causes the PL sender to enter the ERROR state.

SEARCHING: The SEARCHING state is the main probing state. This state is entered when probing for the BASE_PLPMTU completes.

Each time a probe packet is acknowledged, the PROBE_COUNT is set to zero, the PLPMTU is set to the PROBED_SIZE, and then the PROBED_SIZE is increased using the search algorithm (as described in Section 5.3).

When a probe packet is sent and not acknowledged within the period of the PROBE_TIMER, the PROBE_COUNT is incremented, and a new probe packet is transmitted.

The state is exited to enter SEARCH_COMPLETE when the PROBE_COUNT reaches MAX_PROBES, a validated PTB is received that corresponds to the last successfully probed size (PL_PTB_SIZE = PLPMTU), or a probe of size MAX_PLPMTU is acknowledged (PLPMTU = MAX_PLPMTU).

When a black hole is detected in the SEARCHING state, this causes the PL sender to enter the BASE state.

SEARCH_COMPLETE: The SEARCH_COMPLETE state indicates that a search has completed. This is the normal maintenance state, where the PL is not probing to update the PLPMTU. DPLPMTUD remains in this state until either the PMTU_RAISE_TIMER expires or a black hole is detected.

When DPLPMTUD uses an unacknowledged PL and is in the SEARCH_COMPLETE state, a CONFIRMATION_TIMER periodically resets the PROBE_COUNT and schedules a probe packet with the size of the PLPMTU. If MAX_PROBES successive PLPMTUD-sized probes fail to be acknowledged, the method enters the BASE state. When used with an acknowledged PL (e.g., SCTP), DPLPMTUD SHOULD NOT continue to generate PLPMTU probes in this state.

ERROR: The ERROR state represents the case where either the network path is not known to support a PLPMTU of at least the BASE_PLPMTU size or when there is contradictory information about the network path that would otherwise result in excessive variation in the MPS signaled to the higher layer. The state implements a method to mitigate oscillation in the state-event engine. It signals a conservative value of the MPS to the higher layer by the PL. The state is exited when packet probes no longer detect the error. The PL sender then enters the SEARCHING state.

Implementations are permitted to enable endpoint fragmentation if the DPLPMTUD is unable to validate MIN_PLPMTU within PROBE_COUNT probes. If DPLPMTUD is unable to validate MIN_PLPMTU, the implementation will transition to the DISABLED state.

Note: MIN_PLPMTU could be identical to BASE_PLPMTU, simplifying the actions in this state.

5.3. Search to Increase the PLPMTU

This section describes the algorithms used by DPLPMTUD to search for a larger PLPMTU.

5.3.1. Probing for a Larger PLPMTU

Implementations use a search algorithm across the search range to determine whether a larger PLPMTU can be supported across a network path.

The method discovers the search range by confirming the minimum PLPMTU and then using the probe method to select a PROBED_SIZE less than or equal to MAX_PLPMTU. MAX_PLPMTU is the minimum of the local MTU and EMTU_R (when this is learned from the remote endpoint). The MAX_PLPMTU MAY be reduced by an application that sets a maximum to the size of datagrams it will send.

The PROBE_COUNT is initialized to zero when the first probe with a size greater than or equal to PLPMTU is sent. Each probe packet successfully sent to the remote peer is confirmed by acknowledgment at the PL (see Section 4.1).

Each time a probe packet is sent to the destination, the PROBE_TIMER is started. The timer is canceled when the PL receives acknowledgment that the probe packet has been successfully sent across the path (Section 4.1). This confirms that the PROBED_SIZE is supported, and the PROBED_SIZE value is then assigned to the PLPMTU. The search algorithm can continue to send subsequent probe packets of an increasing size.

If the timer expires before a probe packet is acknowledged, the probe has failed to confirm the PROBED_SIZE. Each time the PROBE_TIMER expires, the PROBE_COUNT is incremented, the PROBE_TIMER is reinitialized, and a new probe of the same size or any other size (determined by the search algorithm) can be sent. The maximum number of consecutive failed probes is configured (MAX_PROBES). If the value of the PROBE_COUNT reaches MAX_PROBES, probing will stop, and the PL sender enters the SEARCH_COMPLETE state.

5.3.2. Selection of Probe Sizes

The search algorithm determines a minimum useful gain in PLPMTU. It would not be constructive for a PL sender to attempt to probe for all sizes. This would incur unnecessary load on the path. Implementations SHOULD select the set of probe packet sizes to maximize the gain in PLPMTU from each search step.

Implementations could optimize the search procedure by selecting step sizes from a table of common PMTU sizes. When selecting the appropriate next size to search, an implementer ought to also consider that there can be common sizes of MPS that applications seek to use, and there could be common sizes of MTU used within the network.

5.3.3. Resilience to Inconsistent Path Information

A decision to increase the PLPMTU needs to be resilient to the possibility that information learned about the network path is inconsistent. A path is inconsistent when, for example, probe packets are lost due to other reasons (i.e., not packet size) or due to frequent path changes. Frequent path changes could occur by unexpected "flapping" -- where some packets from a flow pass along one path, but other packets follow a different path with different properties.

A PL sender is able to detect inconsistency either from the sequence of PLPMTU probes that are acknowledged or from the sequence of PTB messages that it receives. When inconsistent path information is detected, a PL sender could use an alternate search mode that clamps the offered MPS to a smaller value for a period of time. This avoids unnecessary loss of packets.

5.4. Robustness to Inconsistent Paths

Some paths could be unable to sustain packets of the BASE_PLPMTU size. The Error State could be implemented to provide robustness to such paths. This allows fallback to a smaller than desired PLPMTU rather than suffer connectivity failure. This could utilize methods such as endpoint IP fragmentation to enable the PL sender to communicate using packets smaller than the BASE_PLPMTU.

6. Specification of Protocol-Specific Methods

DPLPMTUD requires protocol-specific details to be specified for each PL that is used.

The first subsection provides guidance on how to implement the DPLPMTUD method as a part of an application using UDP or UDP-Lite. The guidance also applies to other datagram services that do not include a specific transport protocol (such as a tunnel encapsulation). The following subsections describe how DPLPMTUD can be implemented as a part of the transport service, allowing applications using the service to benefit from discovery of the PLPMTU without themselves needing to implement this method when using SCTP and QUIC.

6.1. Application Support for DPLPMTUD with UDP or UDP-Lite

The current specifications of UDP [RFC0768] and UDP-Lite [RFC3828] do not define a method in the RFC series that supports PLPMTUD. In particular, the UDP transport does not provide the transport features needed to implement datagram PLPMTUD.

The DPLPMTUD method can be implemented as a part of an application built directly or indirectly on UDP or UDP-Lite but relies on higher-layer protocol features to implement the method [BCP145].

Some primitives used by DPLPMTUD might not be available via the Datagram API (e.g., the ability to access the PLPMTU from the IP-layer cache or to interpret received PTB messages).

In addition, it is recommended that PMTU discovery is not performed by multiple protocol layers. An application SHOULD avoid using DPLPMTUD when the underlying transport system provides this capability. A common method for managing the PLPMTU has benefits, both in the ability to share state between different processes and in opportunities to coordinate probing for different PL instances.

6.1.1. Application Request

An application needs an application-layer protocol mechanism (such as a message acknowledgment method) that solicits a response from a destination endpoint. The method SHOULD allow the sender to check the value returned in the response to provide additional protection from off-path insertion of data [BCP145]. Suitable methods include a parameter known only to the two endpoints, such as a session ID or initialized sequence number.

6.1.2. Application Response

An application needs an application-layer protocol mechanism to communicate the response from the destination endpoint. This response could indicate successful reception of the probe across the path but could also indicate that some (or all packets) have failed to reach the destination.

6.1.3. Sending Application Probe Packets

A probe packet can carry an application data block, but the successful transmission of this data is at risk when used for probing. Some applications might prefer to use a probe packet that does not carry an application data block to avoid disruption of data transfer.

6.1.4. Initial Connectivity

An application that does not have other higher-layer information confirming connectivity with the remote peer SHOULD implement a connectivity mechanism using acknowledged probe packets before entering the BASE state.

6.1.5. Validating the Path

An application that does not have other higher-layer information confirming correct delivery of datagrams SHOULD implement the CONFIRMATION_TIMER to periodically send probe packets while in the SEARCH_COMPLETE state.

6.1.6. Handling of PTB Messages

An application that is able and wishes to receive PTB messages MUST perform ICMP validation as specified in Section 5.2 of [BCP145]. This requires that the application checks each received PTB message to validate that it was is received in response to transmitted traffic and that the reported PL_PTB_SIZE is less than the current probed size (see Section 4.6.2). A validated PTB message MAY be used as input to the DPLPMTUD algorithm but MUST NOT be used directly to set the PLPMTU.

6.2. DPLPMTUD for SCTP

Section 10.2 of [RFC4821] specifies a recommended PLPMTUD probing method for SCTP, and Section 7.3 of [RFC4960] recommends an endpoint apply the techniques in RFC 4821 on a per-destination-address basis. The specification for DPLPMTUD continues the practice of using the PL to discover the PMTU but updates RFC4960 with a recommendation to use the method specified in this document: The RECOMMENDED method for generating probes is to add a chunk consisting only of padding to an SCTP message. The PAD chunk defined in [RFC4820] SHOULD be attached to a minimum-length HEARTBEAT (HB) chunk to build a probe packet. This enables probing without affecting the transfer of user messages and without being limited by congestion control or flow control. This is preferred to using DATA chunks (with padding as required) as path probes.

Section 6.9 of [RFC4960] describes dividing the user messages into DATA chunks sent by the PL when using SCTP. This notes that once an SCTP message has been sent, it cannot be resegmented. [RFC4960] describes the method for retransmitting DATA chunks when the MPS has been reduced, and Section 6.9 of [RFC4960] describes use of IP fragmentation for this case. This is unchanged by this document.

6.2.1. SCTP/IPv4 and SCTP/IPv6

6.2.1.1. Initial Connectivity

The base protocol is specified in [RFC4960]. This provides an acknowledged PL. A sender can therefore enter the BASE state as soon as connectivity has been confirmed.

6.2.1.2. Sending SCTP Probe Packets

Probe packets consist of an SCTP common header followed by a HEARTBEAT chunk and a PAD chunk. The PAD chunk is used to control the length of the probe packet. The HEARTBEAT chunk is used to trigger the sending of a HEARTBEAT ACK chunk. The reception of the HEARTBEAT ACK chunk acknowledges reception of a successful probe. A successful probe updates the association and path counters, but an unsuccessful probe is discounted (assumed to be a result of choosing too large a PLPMTU).

The SCTP sender needs to be able to determine the total size of a probe packet. The HEARTBEAT chunk could carry a Heartbeat Information parameter that includes, besides the information suggested in [RFC4960], the probe size to help an implementation associate a HEARTBEAT ACK with the size of probe that was sent. The sender could also use other methods, such as sending a nonce and verifying the information returned also contains the corresponding nonce. The length of the PAD chunk is computed by reducing the probing size by the size of the SCTP common header and the HEARTBEAT chunk. The payload of the PAD chunk contains arbitrary data. When transmitted at the IP layer, the PMTU size also includes the IPv4 or IPv6 header(s).

Probing can start directly after the PL handshake; this can be done before data is sent. Assuming this behavior (i.e., the PMTU is smaller than or equal to the interface MTU), this process will take several round-trip time periods, dependent on the number of DPLPMTUD probes sent. The Heartbeat timer can be used to implement the PROBE_TIMER.

6.2.1.3. Validating the Path with SCTP

Since SCTP provides an acknowledged PL, a sender MUST NOT implement the CONFIRMATION_TIMER while in the SEARCH_COMPLETE state.

6.2.1.4. PTB Message Handling by SCTP

Normal ICMP validation MUST be performed as specified in Appendix C of [RFC4960]. This requires that the first 8 bytes of the SCTP common header are quoted in the payload of the PTB message, which can be the case for ICMPv4 and is normally the case for ICMPv6.

When a PTB message has been validated, the PL_PTB_SIZE calculated from the PTB_SIZE reported in the PTB message SHOULD be used with the DPLPMTUD algorithm, provided that the reported PL_PTB_SIZE is less than the current probe size (see Section 4.6).

6.2.2. DPLPMTUD for SCTP/UDP

The UDP encapsulation of SCTP is specified in [RFC6951].

This specification updates the reference to RFC 4821 in Section 5.6 of RFC 6951 to refer to this document (RFC 8899). RFC 6951 is updated by the addition of the following sentence at the end of Section 5.6:

> The RECOMMENDED method for determining the MTU of the path is specified in RFC 8899.

6.2.2.1. Initial Connectivity

A sender can enter the BASE state as soon as SCTP connectivity has been confirmed.

6.2.2.2. Sending SCTP/UDP Probe Packets

Packet probing can be performed as specified in Section 6.2.1.2. The size of the probe packet includes the 8 bytes of UDP header. This has to be considered when filling the probe packet with the PAD chunk.

6.2.2.3. Validating the Path with SCTP/UDP

SCTP provides an acknowledged PL; therefore, a sender does not implement the CONFIRMATION_TIMER while in the SEARCH_COMPLETE state.

6.2.2.4. Handling of PTB Messages by SCTP/UDP

ICMP validation MUST be performed for PTB messages as specified in Appendix C of [RFC4960]. This requires that the first 8 bytes of the SCTP common header are contained in the PTB message, which can be the case for ICMPv4 (but note the UDP header also consumes a part of the quoted packet header) and is normally the case for ICMPv6. When the validation is completed, the PL_PTB_SIZE calculated from the PTB_SIZE in the PTB message SHOULD be used with the DPLPMTUD providing that the reported PL_PTB_SIZE is less than the current probe size.

6.2.3. DPLPMTUD for SCTP/DTLS

The Datagram Transport Layer Security (DTLS) encapsulation of SCTP is specified in [RFC8261]. This is used for data channels in WebRTC implementations. This specification updates the reference to RFC 4821 in Section 5 of RFC 8261 to refer to this document (RFC 8899).

6.2.3.1. Initial Connectivity

A sender can enter the BASE state as soon as SCTP connectivity has been confirmed.

6.2.3.2. Sending SCTP/DTLS Probe Packets

Packet probing can be done as specified in Section 6.2.1.2. The maximum payload is reduced by the size of the DTLS headers, which has to be considered when filling the PAD chunk. The size of the probe packet includes the DTLS PL headers. This has to be considered when filling the probe packet with the PAD chunk.

6.2.3.3. Validating the Path with SCTP/DTLS

Since SCTP provides an acknowledged PL, a sender MUST NOT implement the CONFIRMATION_TIMER while in the SEARCH_COMPLETE state.

6.2.3.4. Handling of PTB Messages by SCTP/DTLS

[RFC4960] does not specify a way to validate SCTP/DTLS ICMP message payload and neither does this document. This can prevent processing of PTB messages at the PL.

6.3. DPLPMTUD for QUIC

QUIC [QUIC] is a UDP-based PL that provides reception feedback. The UDP payload includes a QUIC packet header, a protected payload, and any authentication fields. It supports padding and packet coalescence that can be used to construct probe packets. From the perspective of DPLPMTUD, QUIC can function as an acknowledged PL. [QUIC] describes the method for using DPLPMTUD with QUIC packets.

7. IANA Considerations

This document has no IANA actions.

8. Security Considerations

The security considerations for the use of UDP and SCTP are provided in the referenced RFCs.

To avoid excessive load, the interval between individual probe packets MUST be at least one RTT, and the interval between rounds of probing is determined by the PMTU_RAISE_TIMER.

A PL sender needs to ensure that the method used to confirm reception of probe packets protects from off-path attackers injecting packets into the path. This protection is provided in IETF-defined protocols (e.g., TCP, SCTP) using a randomly initialized sequence number. A description of one way to do this when using UDP is provided in Section 5.1 of [BCP145]).

There are cases where ICMP Packet Too Big (PTB) messages are not delivered due to policy, configuration, or equipment design (see Section 1.1). This method therefore does not rely upon PTB messages being received but is able to utilize these when they are received by the sender. PTB messages could potentially be used to cause a node to inappropriately reduce the PLPMTU. A node supporting DPLPMTUD MUST therefore appropriately validate the payload of PTB messages to ensure these are received in response to transmitted traffic (i.e., a reported error condition that corresponds to a datagram actually sent by the path layer, see Section 4.6.1).

An on-path attacker able to create a PTB message could forge PTB messages that include a valid quoted IP packet. Such an attack could be used to drive down the PLPMTU. An on-path device could similarly force a reduction of the PLPMTU by implementing a policy that drops packets larger than a configured size. There are two ways this method can be mitigated against such attacks: first, by ensuring that a PL sender never reduces the PLPMTU below the base size solely in response to receiving a PTB message. This is achieved by first entering the BASE state when such a message is received. Second, the design does not require processing of PTB messages; a PL sender could therefore suspend processing of PTB messages (e.g., in a robustness mode after detecting that subsequent probes actually confirm that a size larger than the PTB_SIZE is supported by a path).

Parsing the quoted packet inside a PTB message can introduce additional per-packet processing at the PL sender. This processing SHOULD be limited to avoid a denial-of-service attack when arbitrary headers are included. Rate-limiting the processing could result in PTB messages not being received by a PL; however, the DPLPMTUD method is robust to such loss.

The successful processing of an ICMP message can trigger a probe when the reported PTB size is valid, but this does not directly update the PLPMTU for the path. This prevents a message attempting to black hole data by indicating a size larger than supported by the path.

It is possible that the information about a path is not stable. This could be a result of forwarding across more than one path that has a different actual PMTU or a single path presents a varying PMTU. The design of a PLPMTUD implementation SHOULD consider how to mitigate the effects of varying path information. One possible mitigation is to provide robustness (see Section 5.4) in the method that avoids oscillation in the MPS.

DPLPMTUD methods can introduce padding data to inflate the length of the datagram to the total size required for a probe packet. The total size of a probe packet includes all headers and padding added to the payload data being sent (e.g., including security-related fields such as an AEAD tag and TLS record layer padding). The value of the padding data does not influence the DPLPMTUD search algorithm, and therefore needs to be set consistent with the policy of the PL.

If a PL can make use of cryptographic confidentiality or data-integrity mechanisms, then the design ought to avoid adding anything (e.g., padding) to DPLPMTUD probe packets that is not also protected by those cryptographic mechanisms.

9. References

9.1. Normative References

[BCP145] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, March 2017, <https://www.rfc-editor.org/info/bcp145>.

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <https://www.rfc-editor.org/info/rfc768>.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <https://www.rfc-editor.org/info/rfc791>.

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, DOI 10.17487/RFC1191, November 1990, <https://www.rfc-editor.org/info/rfc1191>.

[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>.

[RFC3828] Larzon, L-A., Degermark, M., Pink, S., Jonsson, L-E., Ed., and G. Fairhurst, Ed., "The Lightweight User Datagram Protocol (UDP-Lite)", RFC 3828, DOI 10.17487/RFC3828, July 2004, <https://www.rfc-editor.org/info/rfc3828>.

[RFC4820] Tuexen, M., Stewart, R., and P. Lei, "Padding Chunk and Parameter for the Stream Control Transmission Protocol (SCTP)", RFC 4820, DOI 10.17487/RFC4820, March 2007, <https://www.rfc-editor.org/info/rfc4820>.

[RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <https://www.rfc-editor.org/info/rfc4960>.

[RFC6951] Tuexen, M. and R. Stewart, "UDP Encapsulation of Stream Control Transmission Protocol (SCTP) Packets for End-Host to End-Host Communication", RFC 6951, DOI 10.17487/RFC6951, May 2013, <https://www.rfc-editor.org/info/rfc6951>.

[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/info/rfc8174>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC8201] McCann, J., Deering, S., Mogul, J., and R. Hinden, Ed., "Path MTU Discovery for IP version 6", STD 87, RFC 8201, DOI 10.17487/RFC8201, July 2017,

<https://www.rfc-editor.org/info/rfc8201>.

[RFC8261] Tuexen, M., Stewart, R., Jesup, R., and S. Loreto, "Datagram Transport Layer Security (DTLS) Encapsulation of SCTP Packets", RFC 8261, DOI 10.17487/RFC8261, November 2017, <https://www.rfc-editor.org/info/rfc8261>.

9.2. Informative References

[QUIC] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", Work in Progress, Internet-Draft, draft-ietf-quic-transport-29, 10 June 2020, <https://tools.ietf.org/html/draft-ietf-quic-transport-29>.

[RFC0792] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, DOI 10.17487/RFC0792, September 1981, <https://www.rfc-editor.org/info/rfc792>.

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <https://www.rfc-editor.org/info/rfc1122>.

[RFC1812] Baker, F., Ed., "Requirements for IP Version 4 Routers", RFC 1812, DOI 10.17487/RFC1812, June 1995, <https://www.rfc-editor.org/info/rfc1812>.

[RFC2923] Lahey, K., "TCP Problems with Path MTU Discovery", RFC 2923, DOI 10.17487/RFC2923, September 2000, <https://www.rfc-editor.org/info/rfc2923>.

[RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, DOI 10.17487/RFC4340, March 2006, <https://www.rfc-editor.org/info/rfc4340>.

[RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <https://www.rfc-editor.org/info/rfc4443>.

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, DOI 10.17487/RFC4821, March 2007, <https://www.rfc-editor.org/info/rfc4821>.

[RFC4890] Davies, E. and J. Mohacsi, "Recommendations for Filtering ICMPv6 Messages in Firewalls", RFC 4890, DOI 10.17487/RFC4890, May 2007, <https://www.rfc-editor.org/info/rfc4890>.

[RFC5508] Srisuresh, P., Ford, B., Sivakumar, S., and S. Guha, "NAT Behavioral Requirements for ICMP", BCP 148, RFC 5508, DOI 10.17487/RFC5508, April 2009, <https://www.rfc-editor.org/info/rfc5508>.

[RFC8900] Bonica, R., Baker, F., Huston, G., Hinden, R., Troan, O., and F. Gont, "IP Fragmentation Considered Fragile", RFC 8900, BCP 230, September 2020, <https://www.rfc-editor.org/info/rfc8900>.

[TUNNELS] Touch, J. and M. Townsley, "IP Tunnels in the Internet Architecture", Work in Progress, Internet-Draft, draft-ietf-intarea-tunnels-10, 12 September 2019, <https://tools.ietf.org/html/draft-ietf-intarea-tunnels-10>.

Acknowledgments

This work was partially funded by the European Union Horizon 2020 Research and Innovation Programme under grant agreement No. 644334, "A New, Evolutive API and Transport-Layer Architecture for the Internet" (NEAT). The views expressed are solely those of the author(s).

Thanks to all who have commented or contributed, the TSVWG and QUIC working groups, and Mathew Calder and Julius Flohr for providing early implementations.

Authors' Addresses

Godred Fairhurst
University of Aberdeen
School of Engineering
Fraser Noble Building
Aberdeen
AB24 3UE
United Kingdom

Email: gorry@erg.abdn.ac.uk

Tom Jones
University of Aberdeen
School of Engineering
Fraser Noble Building
Aberdeen
AB24 3UE
United Kingdom

Email: tom@erg.abdn.ac.uk

Michael Tüxen
Münster University of Applied Sciences
Stegerwaldstrasse 39
48565 Steinfurt
Germany

Email: tuexen@fh-muenster.de

Irene Rüngeler
Münster University of Applied Sciences
Stegerwaldstrasse 39
48565 Steinfurt
Germany

Email: i.ruengeler@fh-muenster.de

Timo Völker
Münster University of Applied Sciences
Stegerwaldstrasse 39
48565 Steinfurt
Germany

Email: timo.voelker@fh-muenster.de