Internet Engineering Task Force
Internet-Draft
Updates: 4821 (if approved)
Intended status: Standards Track
Expires: April 5, 2019

G. Fairhurst
T. Jones
University of Aberdeen
M. Tuexen
I. Ruengeler
Muenster University of Applied Sciences

October 02, 2018

Packetization Layer Path MTU Discovery for Datagram Transports
draft-ietf-tsvwg-datagram-plpmtud-05

Abstract

This document describes a robust method for Path MTU Discovery (PMTUD) for datagram Packetization Layers (PLs). The document describes an extension to RFC 1191 and RFC 8201, which specifies ICMP-based Path MTU Discovery for IPv4 and IPv6. The method 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 network black hole (where packets are discarded, and no ICMP message is received). The method can also probe a network path with progressively larger packets to find whether the maximum packet size can be increased. This allows a sender to determine an appropriate packet size, providing functionally for datagram transports that is equivalent to the Packetization layer PMTUD specification for TCP, specified in RFC 4821.

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

When published, this specification updates RFC 4821.

Status of This Memo

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 April 5, 2019.

Copyright Notice

Table of Contents

1. Introduction

The IETF has specified datagram transport using UDP, SCTP, and 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 may 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]). The term PTB message is applied to both IPv4 ICMP Unreachable messages (Type 3) that carry the error Fragmentation Needed (Type 3, Code 4) and ICMPv6 packet too big messages (Type 2). When a sender receives a PTB message, it reduces the effective MTU to the value reported in the PTB message (in this document called the PTB_SIZE). A method from time-to-time increases the packet size in 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 an attempt to send a packet larger than current effective PMTU fails with an error code. Applications are sometimes provided with a primitive to let them read the maximum packet size, 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 sent with this size, or larger, are silently discarded without the sender receiving ICMP PTB messages). This could arise when the PTB messages are not delivered back to the sender for some reason [RFC2923]). For example, ICMP messages are increasingly filtered by middleboxes (including firewalls) [RFC4890]. A stateful firewall could be configured with a policy to block incoming ICMP messages, which would prevent reception of PTB messages to endpoints behind this firewall. Other examples include cases where PTB messages are not correctly processed/generated by tunnel endpoints.

Another failure could result if a node that is not on the network path sends a PTB message that attempts to force the sender to change the effective PMTU [RFC8201]. A sender can protect itself from reacting to such messages by utilising 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 validation of the PTB message is not possible include:

- When the router issuing the ICMP message is acting on a tunneled packet, the ICMP message will be directed to the tunnel endpoint. This tunnel endpoint is responsible for forwarding the ICMP message and also processing the quoted packet within the payload field to remove the effect of the tunnel, and return a correctly formatted ICMP message to the sender. Failure to do appropriate processing therefore results in black-holing.

- When a router issuing the ICMP message implements RFC 792 [RFC0792], it is only required to include (quote) the first 64 bits of the IP payload of the packet within the ICMP payload. This could be insufficient to perform the tunnel processing described in the previous bullet. Even if the decapsulated message is processed by the tunnel endpoint, there could be insufficient bytes remaining for the sender to interpret the quoted transport information. RFC 1812 [RFC1812] requires routers to return the full packet if possible. This can result in black-holing when used the path includes tunnels.

- When a router issuing the ICMP message quotes a packet with an encrypted transport, it may lack sufficient context to determine the original transport header.

- Even when the PTB message includes sufficient bytes of the quoted packet, the network layer could lack sufficient context to validate the ICMP message, because this 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).

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 Maximum Packet Size (MPS). This function is often performed by a transport protocol, 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] 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 PMTU discovery with TCP.

It uses a general strategy where the PL sends probe packets to search for the largest size of unfragmented datagram that can be sent over a network path. The probe packets are sent with a progressively 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 no response is received to a probe packet, the method reduces the probe size. This PLPMTU is used to set the application MPS.

PLPMTUD introduces flexibility in the implementation of PMTU discovery. At one extreme, it can be configured to only perform PTB black hole detection and recovery to increase the robustness of Classical PMTUD, or at the other extreme, all PTB processing can be disabled and PLPMTUD can completely replace Classical PMTUD.

PLPMTUD can also include additional consistency checks without increasing the risk of increased black-holing. For instance,the information available at the PL, or higher layers, makes PTB validation more straight forward.

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 described relies on 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 utilise ICMP PTB messages when these received messages are made available to the PL.

The UDP Usage Guidelines [RFC8085] state "an application SHOULD either use the Path MTU information provided by the IP layer or implement Path MTU Discovery (PMTUD)", but does not provide a mechanism for discovering the largest size of unfragmented datagram that can be used on a network path. Prior to this document, PLPMTUD had not been specified for UDP.

Section 10.2 of [RFC4821] recommends a PLPMTUD probing method for the Stream Control Transport Protocol (SCTP). SCTP utilises heartbeat messages as probe packets, but RFC4821 does not provide a complete specification. The present document provides the details to complete that 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 could be used with DCCP.

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

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.

Other terminology is directly copied from [RFC4821], and the definitions in [RFC1122].

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 Holed: Packets are Black holed when the sender is unaware that 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 silently discarded by the network, but is not made aware of a change to the path that resulted in a smaller PLPMTU by ICMP messages).

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 datagram 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.

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.

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 by EMTU_R ("Effective MTU to receive").

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 and 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.

MPS: The Maximum Packet Size (MPS) is the largest size of application data block that can be sent across a network path. In DPLPMTUD this quantity is derived from the PLPMTU by taking into consideration the size of the lower protocol layer headers.

MIN_PMTU: The MIN_PMTU is the smallest size of PLPMTU that DPLPTMUD will attempt to use.

Packet: A Packet is the IP header plus the IP payload.

Packetization Layer (PL): The Packetization Layer (PL) is the layer of the network stack that places data into packets and performs transport protocol functions.

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.

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.

PLPMTU: The Packetization Layer PMTU is an estimate of the actual PMTU provided by the DPLPMTUD algorithm.

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.

3. Features Required to Provide Datagram PLPMTUD

TCP PLPMTUD has been defined using standard TCP protocol mechanisms. All of the requirements in [RFC4821] also apply to the use of the technique with a datagram PL. Unlike TCP, some datagram PLs require additional mechanisms to implement PLPMTUD.

There are eight requirements for performing the datagram PLPMTUD method described in this specification:

1. PMTU parameters: A DPLPMTUD sender is RECOMMENDED to provide information about the maximum size of packet that can be transmitted by the sender on the local link (the local Link MTU). It MAY utilize similar information about the receiver when this is supplied (note this could be less than EMTU_R). This avoids implementations trying to send probe packets that can not be transmitted 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).

2. PLPMTU: A datagram application is REQUIRED to be able to choose the size of datagrams sent to the network, up to the PLPMTU, or a smaller value (such as the MPS) derived from this. This value is managed by the DPLPMTUD method. The PLPMTU (specified as the effective PMTU in Section 1 of [RFC1191]) is equivalent to the EMTU_S (specified in [RFC1122]).

3. Probe packets: On request, a DPLPMTUD sender is REQUIRED to be able to transmit a packet larger than the PLMPMTU. This is used to send a probe packet. 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]).

4. 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 originating by the sender, and needs to be performed

before the PLPMTU discovery method reacts to the PTB message. When the PTB_SIZE is indicated in the PTB message, this MAY be used by DPLPMTUD to reduce the probe size but MUST NOT be used to increase the PLPMTU ([RFC8201]). This validation SHOULD utilise information that can not be simply determined by an off-path attacker, for example, by checking the value of a protocol header field known only to the two PL endpoints. (Some datagram applications use well-known source and destination ports and therefore this check needs to rely on other information.)

5. 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. The mechanism needs to be robust to the possibility that packets could be significantly delayed along a network path. The local PL endpoint at the sending node is REQUIRED to pass this feedback to the sender-side DPLPMTUD method.

6. Probing and congestion control: The isolated loss of a probe packet SHOULD NOT be treated as an indication of congestion and its loss SHOULD NOT directly trigger a congestion control reaction [RFC4821].

7. Probe loss recovery: If the data block carried by a probe packet needs to be sent reliably, the PL (or layers above) are REQUIRED to arrange any retransmission/repair of any resulting loss. This method is REQUIRED to be robust in the case where probe packets are lost due to other reasons (including link transmission error, congestion). The DPLPMTUD sender treats isolated loss of a probe packet (with or without an PTB message) as a potential indication of a PMTU limit for the path, but not as an indication of congestion, see Paragraph 6.

8. Shared PLPMTU state: The PLPMTU value could also be stored with the corresponding entry in the destination 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 of [RFC4821]) SHOULD be notified to make use of the new MTU and make the required congestion control adjustments". Such methods MUST be robust to the wide variety of underlying network forwarding behaviours, PLPMTU adjustments based on shared PLPMTU values should be incorporated in the search algorithms. 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:

- o MPS: A method is REQUIRED to signal an appropriate MPS to the higher layer using the PL. The value of the MPS can change following a change to the path. It is RECOMMENDED that methods avoid forcing an application to use an arbitrary small MPS (PLPMTU) for transmission while the method is searching for the currently supported PLPMTU. Datagram PLs do not necessarily support fragmentation of PDUs larger than the PLPMTU. A reduced MPS can adversely impact the performance of a datagram application.

- o 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.

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

- o When to probe: It is RECOMMENDED that methods determine whether the path capacity has increased since it last measured the path. This determines when the path should again be probed.

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 needs to construct 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 utilise 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) could alternatively prefer to generate a probe packet by extending a control message with padding data.

A receiver needs 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 listed in order of preference:

Probing using padding data: A probe packet that contains only control information together with any padding, which is needed to be inflated to the size required for 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 required for the probe packet. If the application/transport needs protection from the loss of this probe packet, the application/transport could perform transport-layer retransmission/repair of the data block (e.g., by retransmission after loss is detected or by duplicating the data block in a datagram without the padding data).

Probing using application data: A probe packet that contains a data block supplied by an application that matches the size required for the probe packet. This method requests the application to issue a data block of the desired probe size. If the application/transport needs protection from the loss of an unsuccessful probe packet, the application/transport needs then to perform transport-layer retransmission/repair of the data block (e.g., by retransmission after loss is detected).

A PL that uses a probe packet carrying an application data block, could need to retransmit this application data block if the probe fails. This could need the PL to re-fragment the data block to a smaller packet size that is expected to traverse the end-to-end path (which could utilise endpoint network-layer or PL fragmentation when these are available).

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 Maximum Segment Lifetime, while being robust to reordering and replay of probe response and ICMP 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 and SCTP provide keep-alive/heartbeat features). When supported, this mechanism SHOULD 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 either rely on an application protocol to detect this loss, or make use of an additional transport method such as UDP-Options [I-D.ietf-tsvwg-udp-options].

Section Section 5 specifies this function for a set of IETF-specified protocols.

4.3. Detection of Black Holes

A PL sender needs to reduce the PLPMTU when it discovers the actual PMTU supported by a network path is less than the PLPMTU (i.e. to detect that traffic is being black holed). This can be triggered when a validated PTB message is received, or by another event that indicates the network path no longer sustains the current packet size, such as a loss report from the PL or repeated lack of response to probe packets sent to confirm the PLPMTU. Detection is followed by a reduction of the PLPMTU.

Black Hole detection is performed by periodically sending packet probes of size PLPMTU to verify that a network path still supports the last acknowledged PLPMTU size. There are two ways a DPLPMTUD sender detect that the current PLPMTU is not sustained by the path (i.e., to detect a black hole):

- o A PL can rely upon a mechanisms implemented within the PL protocol 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 PMTU (as in PLPMTUD for TCP [RFC4821]).

- o A PL can use the probing mechanism to send confirmation probe packets of the size of the current PLPMTU and a timer track whether acknowledgments are received (e.g., The number of probe packets sent without receiving an acknowledgement, PROBE_COUNT, becomes greater than the MAX_PROBES). These messages need to be

generated periodically (e.g., using the confirmation timer Section 5.1.1), and should be suppressed when the PL is not actively sending data. Successive loss of probes is an indication that the current path no longer supports the PLPMTU.

When the method detects the current PLPMTU is not supported (a black hole is found), DPLPMTUD sets a lower MPS. The PL then confirms that the updated PLPMTU can be successfully used across the path. This can need the PL to send a probe packet with a size less than the size of the data block generated by an application. In this case, the PL could provide a way to fragment a datagram at the PL, or could instead utilise a control packet with padding.

4.4. 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 PTB_SIZE indicated in the PTB message, the state of the PLPMTUD state machine, and the IP protocol being used.

Section 4.4.1 first 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.

4.4.1. Validation of PTB Messages

A PL that receives a PTB message from a router or middlebox, MUST perform ICMP validation as specified in Section 5.2 of [RFC8085]. This needs the PL to check the protocol information in the quoted payload to validate the message originated from the sending node. This check includes determining the appropriate port and IP information - necessary for the PTB message to be passed to the PL. In addition, the PL SHOULD validate information from the ICMP payload to determine that the quoted packet was sent by the PL. These checks are intended to provide protection from packets that originate from a node that is not on the network path. PTB messages are discarded if they fail to pass these checks, or where there is insufficient ICMP payload to perform the checks

PTB messages that have been validated can be utilised by the DPLPMTUD algorithm. A method that utilises these PTB messages can improve the speed at the which the algorithm detects an appropriate PLPMTU, compared to one that relies solely on probing.

4.4.2. Use of PTB Messages

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

This section provides an informative summary of how PTB messages can be utilised.

Validating PTB Messages:

* A simple implementation is permitted to ignore received PTB messages and therefore the PLPMTU is not updated when a PTB message is received.

* An implementation that supports PTB messages MUST validate messages before they are processed.

MIN_PMTU < PTB_SIZE < BASE_MTU

* A robust PL MAY enter the PROBE_ERROR state for an IPv4 path when the PTB_SIZE reported in the PTB message >= 576B and when this is less than the BASE_MTU.

* A robust PL MAY enter the PROBE_ERROR state for an IPv6 path when the PTB_SIZE reported in the PTB message >= 1280B and when this is less than the BASE_MTU.

PTB_SIZE = PLPMTU

* Transition to SEARCH_COMPLETE.

PTB_SIZE > PROBED_SIZE

* The PTB_SIZE > PROBED_SIZE, inconsistent network signal. These PTB messages ought to be discarded without further processing (the PLPMTU not updated).

* The information could be utilised as an input to trigger enabling a resilience mode.

BASE_PMTU <= PTB_SIZE < PLPMTU

* Black hole detection is triggered and the PLPMTU ought to be set to BASE_PMTU.

   * The PL could use PTB_SIZE reported in the PTB message to initialise a search algorithm.

PLPMTU < PTB_SIZE < PROBED_SIZE

   * The PLPMTU continues to be valid, but the last PROBED_SIZE searched was larger than the actual PMTU.

   * The PLPMTU is not updated.

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

5. Datagram Packetization Layer PMTUD

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

```
  +----------------------+
  |         APP*         |
  +-+-------+----+---+---+
    |       |    |   |
+---+--+ +--+--+ | +-+---+
| QUIC*| |UDPO*| | |SCTP*|
+---+--+ +--+--+ | ++--+-+
    |       |    |  |  |
    +-------+-+  |  |  |
              |  |  |  |
             ++-+--++  |
             | UDP  |  |
             +---+--+  |
                 |     |
  +--------------+-----+-+
  |  Network Interface   |
  +----------------------+
```

Figure 1: 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 is completely transferred across the network path from the sender to the destination.

This section identifies the components needed for implementation, the phases of operation, the state machine and search algorithm.

5.1. DPLPMTUD Components

This section describes components of DPLPMTUD.

5.1.1. Timers

The method utilises 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 be larger than 1 second, and SHOULD be larger than 15 seconds. Guidance on selection of the timer value are provided in section 3.1.1 of the UDP Usage Guidelines [RFC8085].

If the PL has a path Round Trip Time (RTT) estimate and timely acknowledgements the PROBE_TIMER can be derived from the PL RTT estimate.

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

DPLPMTUD SHOULD inhibit sending probe packets when no application data has been sent since the previous probe packet.

CONFIRMATION_TIMER: 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[RFC8085].

DPLPMTUD SHOULD inhibit sending probe packets when no application data has been sent since the previous probe packet.

An implementation could implement the various timers using a single timer process.

5.1.2. Constants

The following constants are defined:

MAX_PROBES: MAX_PROBES is the maximum value of the PROBE_ERROR_COUNTER. The default value of MAX_PROBES is 10.

MIN_PMTU: The MIN_PMTU is smallest allowed probe packet size. For IPv6, this value is 1280 bytes, as specified in [RFC2460]. For IPv4, the minimum value is 68 bytes. (An IPv4 router is required to be able to forward a datagram of 68 octets without further fragmentation. This is the combined size of an IPv4 header and the minimum fragment size of 8 octets. In addition, receivers are required to be able to reassemble fragmented datagrams at least up to 576B, as stated in section 3.3.3 of [RFC1122]))

MAX_PMTU: The MAX_PMTU is the largest size of PLPMTU. This has to be less than or equal to the minimum of the local MTU of the outgoing interface and the destination PMTU for receiving. An application or PL MAY reduce the MAX_PMTU when there is no need to send packets larger than a specific size.

BASE_PMTU: The BASE_PMTU is a configured size expected to work for most paths. The size is equal to or larger than the MIN_PMTU and smaller than the MAX_PMTU. In the case of IPv6, this value is 1280 bytes [RFC2460]. When using IPv4, a size of 1200 bytes is RECOMMENDED.

5.1.3. Variables

This method utilises a set of variables:

PROBED_SIZE: The PROBED_SIZE is the size of the current probe packet. 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 unsuccessful probe packets that have been sent with a size of PROBED_SIZE. The value is initialised to zero when a particular size of PROBED_SIZE is first attempted.

The figure below illustrates the relationship between the packet size constants and variables, in this case when the DPLPMTUD algorithm performs path probing to increase the size of the PLPMTU. The MPS is less than the PLPMTU. A probe packet has been sent of size PROBED_SIZE. When this is acknowledged, the PLPMTU will be raised to PROBED_SIZE allowing the PROBED_SIZE to be increased towards the actual PMTU.

```
        MIN_PMTU                                             PMTU_MAX
          <------------------------------------------------------>
                         |       |    |     |           |
                         V       |    |     |           V
                     BASE_PMTU   V    |     V     Actual PMTU
                                MPS   |  PROBED_SIZE
                                      V
                                    PLPMTU
```

Figure 2: Relationships between probe and packet sizes

5.2. DPLPMTUD Phases

The Datagram PLPMTUD algorithm moves through several phases of operation.

An implementation 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 fail-over decision).

Black hole detection, see Section 4.3 and PTB processing Section 4.4 proceed in parallel with these phases of operation.

```
                        +-------------------+
                        | Path Confirmation +--       Connectivity
                        +--------+----------+  \-----   or BASE_PMTU
                                 |     /\          \/ Confirmation Fails
            Connectivity and     |     |         +-------+
             BASE_PMTU confirmed |      ---------+ Error |
                                 |               +-------+
                                 |   CONFIRMATION_TIMER
                                 |        Fires
                                 \/
+----------------+          +--------------+
| Search Complete|<---------+   Search     |
+----------------+          +--------------+
                Search Algorithm
                   Completes
```

Figure 3: DPLPMTUD Phases

Path Confirmation

* Connectivity is confirmed.

* DPLPMTUD confirms the BASE_PMTU is supported across the network path.

* DPLPMTUD then enters the search phase.

Search

* DPLPMTUD performs probing to increase the PLPMTU.

* DPLPMTUD then enters the search complete or an error phase.

Search Complete

* DPLPMTUD has found a suitable PLPMTU that is supported across the network path.

* Black hole detection will confirm this PLPMTU continues to be supported.

* On a longer time-frame, DPLPMTUD will re-enter the search phase to discover if the PLPMTU can be raised.

Error

* Inconsistent or invalid network signals cause DPLPMTUD to be unable to progress.

* This causes the algorithm to lower the MPS until the path is shown to support the BASE_PMTU, or to suspend DPLPMTUD.

5.2.1. Path Confirmation Phase

DPLPMTUD starts in the Path confirmation phase. Path confirmation is performed in two stages:

1. Connectivity to the remote peer is first confirmed. When a connection-oriented PL is used, this stage is implicit. It is performed as part of the normal PL connection handshake. In contrast, an connectionless PL MUST send an acknowledged probe packet to confirm that the remote peer is reachable.

2. In the second stage, the PL confirms it can successfully send a datagram of the BASE_PMTU size across the current path.

A PL that does not wish to support a network path with a PLPMTU less than BASE_PMTU can simplify the phase into a single step by performing connectivity checks with probes of the BASE_PMTU size.

A PL MAY respond to PTB messages while in this phase, see Section 4.4.

Once path confirmation has completed, DPLPMTUD can advertise an MPS to an upper layer.

If DPLPMTUD fails to complete these tests it enters the PROBE_DISABLED phase, see Section 5.2.6, and ceases using DPLPTMUD.

5.2.2. Search Phase

The search phase utilises a search algorithm in attempt to increase the PLPMTU (see Section 5.4.1). The PL sender increases the MPS each time a packet probe confirms a larger PLPMTU is supported by the path. The algorithm concludes by entering the SEARCH_COMPLETE phase, see Section 5.2.3.

A PL MAY respond to PTB messages while in this phase, using the PTB to advance or terminate the search, see Section 4.4. Similarly black hole detection can terminate the search by entering the PROBE_BASE phase, see Section 5.2.4.

5.2.2.1. Resilience to inconsistent path information

Sometimes a PL sender is able to detect inconsistent results from the sequence of PLPMTU probes that it sends or the sequence of PTB messages that it receives. This could be manifested as excessive fluctuation of the MPS.

When inconsistent path information is detected, a PL sender can enable an alternate search mode that clamps the offered MPS to a smaller value for a period of time. This avoids unnecessary black-holing of packets.

5.2.3. Search Complete Phase

On entry to the search complete phase, the DPLPMTUD sender starts the PMTU_RAISE_TIMER. In this phase, the PLPMTU remains at the value confirmed by the last successful probe packet.

In this phase, the PL MUST periodically confirm that the PLPMTU is still supported by the path. If the PL is designed in a way that is unable to confirm reachability to the destination endpoint after

probing has completed, the method uses a CONFIRMATION_TIMER to periodically repeat a probe packet for the current PLPMTU size.

If the DPLPMTUD sender is unable to confirm reachability for packets with a size of the current PLPMTU (e.g., if the CONFIRMATION_TIMER expires) or the PL signals a lack of reachability, the method exits the phase and enters the PROBE_BASE phase, see Section 5.2.4.

If the PMTU_RAISE_TIMER expires, the DPLPMTUD sender re-enters the Search phase, see Section 5.2.2, and resumes probing for a larger PLPMTU.

Back hole detection can be used in parallel to check that a network path continues to support a previously confirmed PLPMTU. If a black hole is detected the algorithm moves to the PROBE_BASE phase, see Section 5.2.4.

The phase can also exited when a validated PTB message is received (see Section 4.4.1).

5.2.4. PROBE_BASE Phase

This phase is entered when black hole detection or a PTB message indicates that the PLPMTU is not supported by the path.

On entry to this phase, the PLPMTU is set to the BASE_PMTU, and a corresponding reduced MPS is advertised.

PROBED_SIZE is then set to the PLPMTU (i.e., the BASE_PMTU), to confirm this size is supported across the path. If confirmed, DPLPMTUD enters the Search Phase to determine whether the PL sender can use a larger PLPMTU.

If the path cannot be confirmed to support the BASE_PMTU after sending MAX_PROBES, DPLPMTUD moves to the Error phase, see Section 5.2.5.

5.2.5. ERROR Phase

The ERROR phase is entered when there is conflicting or invalid PLPMTU information for the path (e.g. a failure to support the BASE_PMTU). In this phase, the MPS is set to a value less than the BASE_PMTU, but at least the size of the MIN_PMTU.

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_PMTU.

Note: MIN_PMTU may be identical to BASE_PMTU, simplifying the actions in this phase.

If no acknowledgement is received for PROBE_COUNT probes of size MIN_PMTU, the method suspends DPLPMTUD, see Section 5.2.5.

5.2.5.1. Robustness to inconsistent path

Robustness to paths unable to sustain the BASE_PMTU. Some paths could be unable to sustain packets of the BASE_PMTU size. These paths could use an alternate algorithm to implement the PROBE_ERROR phase that allows fallback to a smaller than desired PLPMTU, rather than suffer connectivity failure.

This could also utilise methods such as endpoint IP fragmentation to enable the PL sender to communicate using packets smaller than the BASE_PMTU.

5.2.6. DISABLED Phase

This phase suspends operation of DPLPMTUD. It disables probing for the PLPMTU until action is taken by the PL or application using the PL.

5.3. State Machine

A state machine for DPLPMTUD is depicted in Figure 4. If multihoming is supported, a state machine is needed for each active path.

```
                                             PROBE_TIMER expiry
                                         (PROBE_COUNT = MAX_PROBES)
                           +-------------------+       +---------------+
                           |    PROBE_START    +------>|PROBE_DISABLED|
                           +-------------------+       +---------------+
                                      |                              ^
                                      | Path confirmed               |
                                      v                              |
   MAX_PMTU acked or           +-------------+-+ (PROBE_COUNT        |
   PTB (BASE_PMTU <= +---------| PROBE_SEARCH | |  < MAX_PROBES)      |
     PTB_SIZE        |    +--> +-------------+<+  or Probe acked     |
   <PROBED_SIZE)     |    |           |   ^  |                       |
       or            |    |           |   |  |                       |
   (PROBE_COUNT      |    |           |   |  |((PTB_SIZE <           |
     =MAX_PROBES)    |    |           |   |  |  BASE_PMTU)           |
     +---------------+    |           |   |  | or                    |
     |                    |           |   |  |(PLPMTU < BASE_MTU))   |
     |                    |           |   |  |and (PROBE_COUNT =     |
     |   PMTU_RAISE_TIMER |           |   |  |   MAX_PROBES)         |
     |                    |           |   |  |                       |
     |                    |           |   |   \                      |
     |        +-----------+           |   |    \   Suspend DPLPDMTUD:|
     |        |                       |   |     \                    |
     |        |                       |   |      \---------+         |
     |        |    (PTB_SIZE < PLPMTU)|   |                |         |
     |        |           or          |   | BASE_PMTU      |         |
     |        |   Black hole detected |   | Probe acked    |         |
     v        |                       v   |                v         |
   +----------+----+            +--------------+        +-------------+
   |SEARCH_COMPLETE|----------->|  PROBE_BASE  |<-------| PROBE_ERROR |
   +------+--------+            +--------------+        +-------------+
    /\    |    Black hole detected  ^  | |  BASE_PMTU Probe acked:   ^
     |    |             or          |  | |                           |
     |    |    (PTB_SIZE < PLPMTU)  |  | | Probe BASE_PMTU:          |
     |    |                         |  | | (PROBE_COUNT = MAX_PROBES)|
     |    |                         |  | +---------------------------+
     +----+                         +--+
    Confirmation:                 PROBE_TIMER expiry:
   (PROBE_COUNT < MAX_PROBES)    (PROBE_COUNT < MAX_PROBES)
            or
    PLPMTU Probe acked
```

Figure 4: State machine for Datagram PLPMTUD. Note: Some state changes are not show to simplify the diagram.

The following states are defined:

PROBE_START: The PROBE_START state is the initial state before probing has started. The state confirms connectivity to the remote PL.

The PLPMTU is set to the BASE_PMTU size. Probing ought to start immediately after connection setup to prevent the prevent the loss of user data. PLPMTUD is not performed in this state. The state transitions to PROBE_SEARCH, when a network path has been confirmed, i.e., when a sent packet has been acknowledged on this network path and the BASE_PMTU is confirmed to be supported. If the network path cannot be confirmed this state transitions to PROBE_DISABLED.

PROBE_SEARCH: The PROBE_SEARCH state is the main probing state. This state is entered when probing for the BASE_PMTU was successful.

The PROBE_COUNT is set to zero when the first probe packet is sent for each probe size. Each time a probe packet is acknowledged, the PLPMTU is set to the PROBED_SIZE, and then the PROBED_SIZE is increased using the search algorithm.

When a probe packet is sent and not acknowledged within the period of the PROBE_TIMER, the PROBE_COUNT is incremented and the probe packet is retransmitted. The state is exited when the PROBE_COUNT reaches MAX_PROBES; a PTB message is validated; a probe of size PMTU_MAX is acknowledged or black hole detection is triggered.

SEARCH_COMPLETE: The SEARCH_COMPLETE state indicates a successful end to the PROBE_SEARCH state. DPLPMTUD remains in this state until either the PMTU_RAISE_TIMER expires; a received PTB message is validated; or black hole detection is triggered.

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 the probe packet fails to be acknowledged after MAX_PROBES attempts, the method enters the PROBE_BASE state. When used with an acknowledged PL (e.g., SCTP), DPLPMTUD SHOULD NOT continue to generate PLPMTU probes in this state.

PROBE_BASE: The PROBE_BASE state is used to confirm whether the BASE_PMTU 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 where traffic is black holed while searching for a larger PLPMTU.

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

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

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

PROBE_ERROR: The PROBE_ERROR state represents the case where the network path is not known to support a PLPMTU of at least the BASE_PMTU size. It is entered when either a probe of size BASE_PMTU has not been acknowledged or a validated PTB message indicates a smaller PTB_SIZE smaller than the BASE_PMTU.

On entry, the PROBE_COUNT is set to zero and the PROBED_SIZE is set to the MIN_PMTU size, and the PLPMTU is reset to MIN_PMTU size. In this state, a probe packet is sent, and the PROBE_TIMER is started. The state transitions to the PROBE_SEARCH state when a probe packet is acknowledged of at least size BASE_PMTU. Robust implementations may validate the BASE_PMTU several times before transition to the PROBE_SEARCH.

Implementations are permitted to enable endpoint fragmentation if the DPLPMTUD is unable to validate MIN_PMTU within PROBE_COUNT probes. If DPLPMTUD is unable to validate MIN_PMTU the implementation should transition to PROBE_DISABLED.

PROBE_DISABLED: The PROBE_DISABLED state indicates that connectivity could not be established. DPLPMTUD MUST NOT probe in this state.

Appendix A contains an informative description of key events.

5.4. Search to Increase the PLPMTU

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

5.4.1. 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 PMTU_MAX. PMTU_MAX is the minimum of the local MTU and EMTU_R (learned from the remote endpoint). The PMTU_MAX MAY be reduced by an application that sets a maximum to the size of datagrams it will send.

The PROBE_COUNT is initialised to zero when a probe packet is first sent with a particular size. A timer is used by the search algorithm to trigger the sending of probe packets of size PROBED_SIZE, larger than the PLPMTU. Each probe packet successfully sent to the remote peer is confirmed by acknowledgement 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 cancelled 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 reinitialised, and a probe packet of the same size is retransmitted (the replicated probe improve the resilience to loss). The maximum number of retransmissions for a particular size 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.

5.4.2. Selection of Probe Sizes

The search algorithm needs to determine 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 and has the undesirable effect of slowing the time to reach a more optimal MPS. Implementations SHOULD select the set of probe packet sizes to maximise 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 implementor ought to also consider that there can be common sizes of MPS that applications seek to use.

xxx Author Note: A future version of this section will detail example methods for selecting probe size values, but does not plan to mandate a single method. xxx

5.4.3. 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 (this could happen when probe packets are lost due to other reasons, or some of the packets in a flow are forwarded along a portion of the path that supports a different actual PMTU).

Frequent path changes could occur due to unexpected "flapping" - where some packets from a flow pass along one path, but other packets follow a different path with different properties. DPLPMTUD can be made resilient to these anomalies by introducing hysteresis into the search decision to increase the MPS.

6. Specification of Protocol-Specific Methods

This section specifies protocol-specific details for datagram PLPMTUD for IETF-specified transports.

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 subsection 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.

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 layer 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 [RFC8085].

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

In addition, it is desirable that PMTU discovery is not performed by multiple protocol layers. An application SHOULD avoid implementing DPLPMTUD when the underlying transport system provides this capability. Using a common method for managing the PLPMTU has

benefits, both in the ability to share state between different processes and opportunities to coordinate probing.

6.1.1. Application Request

An application needs an application-layer protocol mechanism (such as a message acknowledgement 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 [RFC8085], suitable methods include a parameter known only to the two endpoints, such as a session ID or initialised 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 may 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 that may carry an application data block, but the successful transmission of this data is at risk when used for probing. Some applications may prefer to use a probe packet that does not carry an application data block to avoid disruption to normal data transfer.

6.1.4. 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.

6.1.5. 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 [RFC8085]. This requires that the application to check each received PTB messages to validate it is received in response to transmitted traffic and that the reported PTB_SIZE is less than the current probed size. 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 with UDP Options

UDP Options[I-D.ietf-tsvwg-udp-options] can supply the additional functionality required to implement DPLPMTUD within the UDP transport service. Implementing DPLPMTU using UDP Options avoids the need for each application to implement the DPLPMTUD method.

Section 5.6 of[I-D.ietf-tsvwg-udp-options] defines the Maximum Segment Size (MSS) option, which allows the local sender to indicate the EMTU_R to the peer. The value received in this option can be used to initialise PMTU_MAX.

UDP Options enables padding to be added to UDP datagrams that are used as Probe Packets. Feedback confirming reception of each Probe Packet is provided by two new UDP Options:

- o The Probe Request Option (Section 6.2.1) is set by a sending PL to solicit a response from a remote endpoint. A four-byte token identifies each request.

- o The Probe Response Option (Section 6.2.2 is generated by the UDP Options receiver in response to reception of a previously received Probe Request Option. Each Probe Response Option echoes a previously received four-byte token.

The token value allows implementations to be distinguish between acknowledgements for initial probe packets and acknowledgements confirming receipt of subsequent probe packets (e.g., travelling along alternate paths with a larger RTT). Each probe packet needs to be uniquely identifiable by the UDP Options sender within the Maximum Segment Lifetime (MSL). The UDP Options sender therefore needs to not recycle token values until they have expired or have been acknowledged. A 4 byte value for the token field provides sufficient space for multiple unique probes to be made within the MSL.

The initial value of the four byte token field SHOULD be assigned to a randomised value, as described in section 5.1 of [RFC8085]) to enhance protection from off-path attacks.

Implementations ought to only send a probe packet with a Request Probe Option when required by their local state machine, i.e., when probing to grow the PLPMTU or to confirm the current PLPMTU. The procedure to handle the loss of a response packet is the responsibility of the sender of the request. Implementations are allowed to track multiple requests and respond to them with a single packet.

A PL needs to determine that the path can still support the size of datagram that the application is currently sending in the DPLPMTUD search_done state (i.e., to detect black-holing of data). One way to achieve this is to send probe packets of size PLPMTU or to utilise a higher-layer method that provides explicit feedback indicating any packet loss. Another possibility is to utilise data packets that carry a Timestamp Option. Reception of a valid timestamp that was echoed by the remote endpoint can be used to infer connectivity. This can provide useful feedback even over paths with asymmetric capacity and/or that carry UDP Option flows that have very asymmetric datagram rates, because an echo of the most recent timestamp still indicates reception of at least one packet of the transmitted size. This is sufficient to confirm there is no black hole.

In contrast, when sending a probe to increase the PLPMTU, a timestamp might be unable to unambiguously identify that a specific probe packet has been received. Timestamp mechanisms cannot be used to confirm the reception of individual probe messages and cannot be used to stimulate a response from the remote peer.

6.2.1. UDP Probe Request Option

The Probe Request Option allows a sending endpoint to solicit a response from a destination endpoint.

The Probe Request Option carries a four byte token set by the sender. This token can be set to a value that is likely to be known only to the sender (and is sent along the end-to-end path). The initial value of the token SHOULD be assigned to a randomised value, as described in section 5.1 of [RFC8085]) to enhance protection from off-path attacks.

The sender needs to then check the value returned in the UDP Probe Response Option. The value of the Token field, uniquely identifies a probe within the maximum segment lifetime.

```
+----------+--------+-----------------+
| Kind=9* | Len=6  |     Token       |
+----------+--------+-----------------+
  1 byte    1 byte       4 bytes

   * To be confirmed by IANA.
```

Figure 5: UDP Probe REQ Option Format

6.2.2. UDP Probe Response Option

The Probe Response Option is generated in response to reception of a previously received Probe Request Option. This response is generated by the UDP Option processing.

The Probe Response Option carries a four byte token field. The Token field associates the response with the Token value carried in the most recently-received Echo Request. The rate of generation of UDP packets carrying a Probe Response Option is expected to be less than once per RTT and SHOULD be rate-limited (see Section 9).

```
+----------+--------+-----------------+
| Kind=10* | Len=6  |      Token      |
+----------+--------+-----------------+
  1 byte     1 byte       4 bytes

  * To be confirmed by IANA.
```

Figure 6: UDP Probe RES Option Format

6.3. DPLPMTUD for SCTP

Section 10.2 of [RFC4821] specifies a recommended PLPMTUD probing method for SCTP. It recommends the use of the PAD chunk, defined in [RFC4820] to be attached to a minimum length HEARTBEAT chunk to build a probe packet. This enables probing without affecting the transfer of user messages and without interfering with congestion control. This is preferred to using DATA chunks (with padding as required) as path probes.

XXX Author Note: Future versions of this document might define a parameter contained in the INIT and INIT ACK chunk to indicate the remote peer MTU to the local peer. However, multihoming makes this a bit complex, so it might not be worth doing. XXX

6.3.1. SCTP/IPv4 and SCTP/IPv6

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

6.3.1.1. 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.

The HEARTBEAT chunk carries a Heartbeat Information parameter which should include, besides the information suggested in [RFC4960], the probe size, which is the size of the complete datagram. The size of the PAD chunk is therefore computed by reducing the probing size by the IPv4 or IPv6 header size, the SCTP common header, the HEARTBEAT request and the PAD chunk header. The payload of the PAD chunk contains arbitrary data.

To avoid fragmentation of retransmitted data, probing starts right after the handshake, before data is sent. Assuming normal behaviour (i.e., the PMTU is smaller than or equal to the interface MTU), this process will take a few round trip time periods depending on the number of PMTU sizes probed. The Heartbeat timer can be used to implement the PROBE_TIMER.

6.3.1.2. 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.

6.3.1.3. 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 PTB_SIZE reported in the PTB message SHOULD be used with the DPLPMTUD algorithm, providing that the reported PTB_SIZE is less than the current probe size.

6.3.2. DPLPMTUD for SCTP/UDP

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

6.3.2.1. Sending SCTP/UDP Probe Packets

Packet probing can be performed as specified in Section 6.3.1.1. The maximum payload is reduced by 8 bytes, which has to be considered when filling the PAD chunk.

6.3.2.2. Validating the Path with SCTP/UDP

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

6.3.2.3. Handling of PTB Messages by SCTP/UDP

Normal 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 PTB_SIZE indicated in the PTB message SHOULD be used with the DPLPMTUD providing that the reported PTB_SIZE is less than the current probe size.

6.3.3. DPLPMTUD for SCTP/DTLS

The Datagram Transport Layer Security (DTLS) encapsulation of SCTP is specified in [RFC8261]. It is used for data channels in WebRTC implementations.

6.3.3.1. Sending SCTP/DTLS Probe Packets

Packet probing can be done as specified in Section 6.3.1.1.

6.3.3.2. 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.

6.3.3.3. Handling of PTB Messages by SCTP/DTLS

It is not possible to perform normal ICMP validation as specified in [RFC4960], since even if the ICMP message payload contains sufficient information, the reflected SCTP common header would be encrypted. Therefore it is not possible to process PTB messages at the PL.

6.4. DPLPMTUD for QUIC

Quick UDP Internet Connection (QUIC) [I-D.ietf-quic-transport] is a UDP-based transport that provides reception feedback.

Section 9.2 of [I-D.ietf-quic-transport] describes the path considerations when sending QUIC packets. It recommends the use of PADDING frames to build the probe packet. This enables probing without affecting the transfer of other QUIC frames.

This provides an acknowledged PL. A sender can therefore enter the PROBE_BASE state as soon as connectivity has been confirmed.

6.4.1. Sending QUIC Probe Packets

A probe packet consists of a QUIC Header and a payload containing only PADDING Frames. PADDING Frames are a single octet (0x00) and several of these can be used to create a probe packet of size PROBED_SIZE. QUIC provides an acknowledged PL. A sender can therefore enter the PROBE_BASE state as soon as connectivity has been confirmed.

The current specification of QUIC sets the following:

- BASE_PMTU: 1200. A QUIC sender needs to pad initial packets to 1200 bytes to confirm the path can support packets of a useful size.

- MIN_PMTU: 1200 bytes. A QUIC sender that determines the PMTU has fallen below 1200 bytes MUST immediately stop sending on the affected path.

6.4.2. Validating the Path with QUIC

QUIC provides an acknowledged PL. A sender therefore MUST NOT implement the CONFIRMATION_TIMER while in the SEARCH_COMPLETE state.

6.4.3. Handling of PTB Messages by QUIC

QUIC operates over the UDP transport, and the guidelines on ICMP validation as specified in Section 5.2 of [RFC8085] therefore apply. Although QUIC does not currently specify a method for validating ICMP responses, it does provide some guidelines to make it harder for an off-path attacker to inject ICMP messages.

- Set the IPv4 Don't Fragment (DF) bit on a small proportion of packets, so that most invalid ICMP messages arrive when there are no DF packets outstanding, and can therefore be identified as spurious.

- Store additional information from the IP or UDP headers from DF packets (for example, the IP ID or UDP checksum) to further authenticate incoming Datagram Too Big messages.

- Any reduction in PMTU due to a report contained in an ICMP packet is provisional until QUIC's loss detection algorithm determines that the packet is actually lost.

XXX The above list was pulled whole from quic-transport - input is invited from QUIC contributors. XXX

7. Acknowledgements

This work was partially funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 644334 (NEAT). The views expressed are solely those of the author(s).

8. IANA Considerations

This memo includes no request to IANA.

XXX If new UDP Options are specified in this document, a request to IANA will be included here. XXX

If there are no requirements for IANA, the section will be removed during conversion into an RFC by the RFC Editor.

9. Security Considerations

The security considerations for the use of UDP and SCTP are provided in the references RFCs. Security guidance for applications using UDP is provided in the UDP Usage Guidelines [RFC8085], specifically the generation of probe packets is regarded as a "Low Data-Volume Application", described in section 3.1.3 of this document. This recommends that sender limits generation of probe packets to an average rate lower than one probe per 3 seconds.

A PL sender needs to ensure that the method used to confirm reception of probe packets offers protection from off-path attackers injecting packets into the path. This protection if provided in IETF-defined protocols (e.g., TCP, SCTP) using a randomly-initialised sequence number. A description of one way to do this when using UDP is provided in section 5.1 of [RFC8085]).

There are cases where 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 utilise 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).

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. 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 PROBE_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).

Parallel forwarding paths SHOULD be considered. Section 5.2.5.1 identifies the need for robustness in the method when the path information may be inconsistent.

A node performing DPLPMTUD could experience conflicting information about the size of supported probe packets. This could occur when there are multiple paths are concurrently in use and these exhibit a different PMTU. If not considered, this could result in data being black holed when the PLPMTU is larger than the smallest PMTU across the current paths.

10. References

10.1. Normative References

[I-D.ietf-quic-transport]
Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", draft-ietf-quic-transport-14 (work in progress), August 2018.

[I-D.ietf-tsvwg-udp-options]
Touch, J., "Transport Options for UDP", draft-ietf-tsvwg-udp-options-05 (work in progress), July 2018.

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

[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>.

[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>.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <https://www.rfc-editor.org/info/rfc2460>.

[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>.

[RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <https://www.rfc-editor.org/info/rfc8085>.

[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>.

[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>.

10.2. Informative References

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

[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>.

[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>.

Appendix A. Event-driven state changes

This appendix contains an informative description of key events:

Path Setup: When a new path is initiated, the state is set to PROBE_START. This sends a probe packet with the size of the BASE_PMTU. As soon as the path is confirmed, the state changes to PROBE_SEARCH.

Arrival of an Acknowledgment: Depending on the probing state, the reaction differs according to Figure 7, which is a simplification of Figure 4 focusing on this event.

```
  +--------------+                                    +----------------+
  | PROBE_START  | --3-------------------------------> | PROBE_DISABLED |
  +--------------+ --4----------------  -------------> +----------------+
                                      \/
  +--------------+                    /\              +--------------+
  | PROBE_ERROR  | -------------------- \ -----------> |  PROBE_BASE  |
  +--------------+ --4--------------/    \            +--------------+
                                          \
  +--------------+ --1 --------            \          +--------------+
  |  PROBE_BASE  |             \        --- \ ------> | PROBE_ERROR  |
  +--------------+ --3--------- \ -----/     \       +--------------+
                                 \            \
  +--------------+                \            -----> +--------------+
  | PROBE_SEARCH | --2---          ------------------> | PROBE_SEARCH |
  +--------------+       \        -------------------> +--------------+
                          \ ---- /
  +---------------+      / \                          +---------------+
  |SEARCH_COMPLETE| -1---   \                         |SEARCH_COMPLETE|
  +---------------+ -5--     ------------------------> +---------------+
                        \
                         \                            +--------------+
                          ---------------------------> |  PROBE_BASE  |
                                                      +--------------+
```

Condition 1: The maximum PMTU size has not yet been reached. Condition 2: The maximum PMTU size has been reached. Condition 3: Probe Timer expires and PROBE_COUNT = MAX_PROBEs. Condition 4: PROBE_ACK received. Condition 5: Black hole detected.

Figure 7: State changes at the arrival of an acknowledgment

Probing timeout: The PROBE_COUNT is initialised to zero each time the value of PROBED_SIZE is changed and when a acknowledgment confirming delivery of a probe packet. The PROBE_TIMER is started each time a probe packet is sent. It is stopped when an acknowledgment arrives that confirms delivery of a probe packet of PROBED_SIZE. If the probe packet is not acknowledged before the PROBE_TIMER expires, the PROBE_COUNT is incremented. When the PROBE_COUNT equals the value MAX_PROBES, the state is changed, otherwise a new probe packet of the same size (PROBED_SIZE) is resent. The state transitions are illustrated in Figure 8. This shows a simplification of Figure 4 with a focus only on this event.

```
  +--------------+                                   +----------------+
  |  PROBE_START | --2------------------------------->| PROBE_DISABLED |
  +--------------+                                   +----------------+

  +--------------+                                   +--------------+
  | PROBE_ERROR  |                 -----------------> | PROBE_ERROR  |
  +--------------+                /                  +--------------+
                                 /
  +--------------+ --2----------/                    +--------------+
  |  PROBE_BASE  | --1------------------------------> |  PROBE_BASE  |
  +--------------+                                   +--------------+

  +--------------+                                   +--------------+
  | PROBE_SEARCH | --1------------------------------> | PROBE_SEARCH |
  +--------------+ --2---------                      +--------------+
                              \
  +---------------+            \                     +---------------+
  |SEARCH_COMPLETE|             -------------------> |SEARCH_COMPLETE|
  +---------------+                                  +---------------+
```

Condition 1: The maximum number of probe packets has not been reached. Condition 2: The maximum number of probe packets has been reached. XXX This diagram has not been validated.

Figure 8: State changes at the expiration of the probe timer

PMTU raise timer timeout: DPLPMTUD periodically sends a probe packet to detect whether a larger PMTU is possible. This probe packet is generated by the PMTU_RAISE_TIMER.

Arrival of a PTB message: The active probing of the path can be supported by the arrival of a PTB message indicating the PTB_SIZE. Two examples are:

1. The PTB_SIZE is between the PLPMTU and the probe that triggered the PTB message.

2. The PTB_SIZE is smaller than the PLPMTU.

In first case, the PROBE_BASE state transitions to the PROBE_ERROR state. In the PROBE_SEARCH state, a new probe packet is sent with the size reported by the PTB message.

In second case, the probing starts again with a value of PROBE_BASE.

Appendix B. Revision Notes

Note to RFC-Editor: please remove this entire section prior to publication.

Individual draft -00:

- Comments and corrections are welcome directly to the authors or via the IETF TSVWG working group mailing list.

- This update is proposed for WG comments.

Individual draft -01:

- Contains the first representation of the algorithm, showing the states and timers

- This update is proposed for WG comments.

Individual draft -02:

- Contains updated representation of the algorithm, and textual corrections.

- The text describing when to set the effective PMTU has not yet been validated by the authors

- To determine security to off-path-attacks: We need to decide whether a received PTB message SHOULD/MUST be validated? The text on how to handle a PTB message indicating a link MTU larger than the probe has yet not been validated by the authors

- No text currently describes how to handle inconsistent results from arbitrary re-routing along different parallel paths

- This update is proposed for WG comments.

Working Group draft -00:

- This draft follows a successful adoption call for TSVWG

- There is still work to complete, please comment on this draft.

Working Group draft -01:

- This draft includes improved introduction.

- o The draft is updated to require ICMP validation prior to accepting PTB messages - this to be confirmed by WG

- o Section added to discuss Selection of Probe Size - methods to be evlauated and recommendations to be considered

- o Section added to align with work proposed in the QUIC WG.

Working Group draft -02:

- o The draft was updated based on feedback from the WG, and a detailed review by Magnus Westerlund.

- o The document updates RFC 4821.

- o Requirements list updated.

- o Added more explicit discussion of a simpler black-hole detection mode.

- o This draft includes reorganisation of the section on IETF protocols.

- o Added more discussion of implementation within an application.

- o Added text on flapping paths.

- o Replaced 'effective MTU' with new term PLPMTU.

Working Group draft -03:

- o Updated figures

- o Added more discussion on blackhole detection

- o Added figure describing just blackhole detection

- o Added figure relating MPS sizes

Working Group draft -04:

- o Described phases and named these consistently.

- o Corrected transition from confirmation directly to the search phase (Base has been checked).

- o Redrawn state diagrams.

- o Renamed BASE_MTU to BASE_PMTU (because it is a base for the PMTU).

- o Clarified Error state.

- o Clarified supsending DPLPMTUD.

- o Verified normative text in requirements section.

- o Removed duplicate text.

- o Changed all text to refer to /packet probe/probe packet/ /validation/verification/ added term /Probe Confirmation/ and clarified BlackHole detection.

Working Group draft -05:

- o Updated security considerations.

- o Feedback after speaking with Joe Touch helped improve UDP-Options description.

Authors' Addresses

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

Email: gorry@erg.abdn.ac.uk

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

Email: tom@erg.abdn.ac.uk

Michael Tuexen
Muenster University of Applied Sciences
Stegerwaldstrasse 39
Stein fart 48565
DE

Email: tuexen@fh-muenster.de

Irene Ruengeler
Muenster University of Applied Sciences
Stegerwaldstrasse 39
Stein fart 48565
DE

Email: i.ruengeler@fh-muenster.de