Path MTU Discovery
draft-ietf-pmtud-method-05

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on April 26, 2006.

Copyright Notice

Copyright (C) The Internet Society (2005).

Abstract

This document describes a robust method for Path MTU Discovery that relies on TCP or some other Packetization Layer to probe an Internet path with progressively larger packets. This method is described as an extension to RFC 1191 and RFC 1981, which specify ICMP based Path MTU Discovery for IP versions 4 and 6, respectively.

The general strategy of the new algorithm is to start with a small MTU and search upward, testing successively larger MTUs by probing
with single packets. If the probe is successfully delivered and satisfies a subsequent verification phase then the MTU is raised. If the probe is lost, it is treated as an MTU limitation and not as a congestion signal.

There are several options for integrating PLPMTUD with classical path MTU discovery. PLPMTUD can be minimally configured to perform ICMP black hole recovery to increase the robustness of classical path MTU discovery, or ICMP processing can be completely disabled, and PLPMTUD can completely replace classical path MTU discovery.

In the latter configuration, PLPMTUD exactly parallels congestion control. An end-to-end transport protocol adjusts non-protocol properties of the data stream (window size or packet size) while using packet losses to deduce the appropriateness of the adjustments. This technique seems to be more philosophically consistent with the end-to-end principle than relying on ICMP messages containing transcribed headers of multiple protocol layers.
Table of Contents

1. Introduction ............................................... 4
   1.1. Revision History ..................................... 4
       1.1.1. Changes since version -04, February 2005 (IETF 62) .... 5
       1.1.2. Changes since version -03, October 2004 (IETF 61) .... 5
       1.1.3. Changes since version -02, July 19th 2004 (IETF 60) .... 5
2. Overview .................................................. 6
3. Terminology ............................................... 9
4. Requirements .............................................. 11
5. Layering .................................................. 12
   5.1. Accounting for header sizes ............................ 13
   5.2. Storing PMTU information ............................... 13
   5.3. Accounting for IPsec ................................... 14
   5.4. Multicast ............................................. 15
6. Common Packetization Properties ................................ 15
   6.1. Mechanism to detect loss ................................ 15
   6.2. Generating probes ...................................... 16
7. Host Fragmentation .......................................... 16
8. The Probing Method ......................................... 17
   8.1. Packet size ranges ..................................... 17
   8.2. Selecting initial values ................................ 18
   8.3. Selecting probe size ................................... 19
   8.4. Probing preconditions .................................. 20
   8.5. Conducting a probe ..................................... 20
   8.6. Response to probe results .............................. 20
       8.6.1. Probe success ....................................... 21
       8.6.2. Probe failure ....................................... 21
       8.6.3. Probe timeout failure ............................... 21
       8.6.4. Probe inconclusive .................................. 22
   8.7. Full stop timeout ..................................... 22
   8.8. MTU verification ...................................... 22
9. Diagnostic Interface ........................................ 23
10. Specific Packetization Layers ............................. 23
   10.1. Probing method using TCP ............................. 24
   10.2. Probing method using SCTP ............................ 24
   10.3. Probing method using IP fragmentation ................ 25
   10.4. Probing method using applications .................... 26
11. References ................................................ 27
   11.1. Normative references .................................. 27
   11.2. Informative references ............................... 28
Appendix A. Security Considerations ......................... 29
Appendix B. IANA Considerations ............................ 29
Appendix C. Acknowledgements .............................. 29
Authors’ Addresses .......................................... 30
Intellectual Property and Copyright Statements ............... 31
1. Introduction

This document describes a method for Packetization Layer Path MTU Discovery (PLPMTUD) which is an extension to existing Path MTU discovery methods as described in RFC 1191 [2] and RFC 1981 [3]. The proper MTU is determined by starting with small packets and probing with successively larger packets. The bulk of the algorithm is implemented above IP, in the transport layer (e.g. TCP) or other "Packetization Protocol" that is responsible for determining packet boundaries.


This document describes methods to discover the path MTU using features of existing protocols. The methods apply to IPv4 and IPv6, and many transport protocols. They do not require cooperation from the lower layers (except that they are consistent about what packet sizes are acceptable) or the far node. Variants in implementations will not cause interoperability problems.

The methods described in this document are carefully designed to maximize robustness in the presence of less than ideal implementations of other protocols or Internet components.

For sake of clarity we uniformly prefer TCP and IPv6 terminology. In the terminology section we also present the analogous IPv4 terms and concepts for the IPv6 terminology. In a few situations we describe specific details that are different between IPv4 and IPv6.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [4].

This draft is a product of the Path MTU Discovery (pmtud) working group of the IETF. Please send comments and suggestions to pmtud@ietf.org. Interim drafts and other useful information will be posted at http://www.psc.edu/~mathis/MTU/pmtud/index.html .

1.1. Revision History

These are all recent substantive changes, in reverse chronological order. This section will be removed prior to publication as an RFC. Note that there are still some missing details that need to be resolved. These are flagged by @@@@. None of the missing details are serious.
1.1.1. Changes since version -04, February 2005 (IETF 62)

General restructuring and rewriting of some sections based on new experience. Relaxed and generalized a lot of over-specified language, e.g., the search strategy description.

Decoupled verification from probing, and relaxed its specification.

Removed all specified changes to ICMP processing. We decided this was out of scope for this particular document.

Changed all language to refer to MTU rather than MPS.

1.1.2. Changes since version -03, October 2004 (IETF 61)

A number of minor style and grammar edits.

1.1.3. Changes since version -02, July 19th 2004 (IETF 60)

Many minor updates throughout the document.

Added a section describing the interactions between PLPMTUD and congestion control.

Removed a difficult to implement requirement for future data to transmit.

Added "IP Fragmentation" and "Application protocol" as Packetization Layers.

Clarified interactions between TCP SACK and MTU.

Updated SCTP section to reflect new probing method using "PAD chunks".

Distilled the protocol specific material into separate subsections for each protocol.

Added a section on common requirements and functions for all Packetization Layers. More accurately characterized the "bidirectional" (and other) requirements of the PL protocol. Updated the search strategy in this new section.

Change "ICMP can’t fragment" and "packet too big" to uniformly use "ICMP PTB message" everywhere.

Added Stanislav Shalunov’s observation that PLPMTUD parallels congestion control.
Better described the range of interoperability with classical pMTUd
in the introduction.

Removed vague language about "not being a protocol" and "excessive
Loss".

Slightly redefined flow: the granularity of PLPMTUD within a path.

Many English NITs and clarifications per Gorry Fairhurst and others.
Passed strict xml2rfc checking.

Add a paragraph encouraging interface MTUs that are the optimal for
the NIC, rather than standard for the media.

Added a revision history section.

2. Overview

This document describes a method for TCP or other packetization
protocols to dynamically discover the MTU of a path without explicit
signals from the network. This method is most efficient when used in
conjunction with the current ICMP based path MTU discovery mechanism
as specified in **RFC1191** and **RFC1981**. When used in such a way, it
eliminates many robustness problems since it does not depend on the
delivery ICMP messages.

These procedures are applicable to TCP and other transport- or
application-level protocols that are responsible for choosing packet
boundaries (e.g. segment sizes) and have an acknowledgment structure
that delivers to the sender accurate and timely indications of which
packets were lost.

The general strategy is for the packetization layer to find an
appropriate path MTU by probing the path with progressively larger
packets. If a probe packet is successfully delivered, then the
effective path MTU is raised to the probe size.

The isolated loss of a probe packet (with or without an ICMP Packet
To Big message) is treated as an indication of an MTU limit, and not
as a congestion indicator. In this case alone, the packetization
protocol is permitted to retransmit any missing data without
adjusting the congestion window.

If there is a timeout or additional packets are lost during the
probing process, the probe is considered to be inconclusive (e.g. the
lost probe does not necessarily indicate that the probe exceeded the
path MTU). Furthermore the losses are treated like any other
congestion indication: window or rate adjustments are mandatory per the relevant congestion control standards [12] Probing can resume after a delay which is determined by the nature of the detected failure.

PLPMTUD uses a searching technique to find the path MTU. Each conclusive probe narrows the MTU search range, either by raising the low limit on a successful probe or lowering the high limit on a failed probe, until the search range converges toward the true path MTU. For most transport layers, it makes sense to abandon the search once the range is narrow enough where the likely gain from picking a larger effective path MTU is smaller than the search overhead to find it.

The most likely (and least serious) PLPMTUD failure is the link experiencing congestion related losses while probing. In this case it is appropriate to retry a probe of the same size as soon as the packetization layer has fully adapted to the congestion and recovered from the losses. In other cases, additional losses or timeouts indicate problems with the link or packetization layer. In these situations it is desirable to use longer delays depending on the severity of the error.

An optional verification phase can be used to detect some situations where raising the MTU raises the packet loss rate. For example, if a link is striped across multiple physical channels with inconsistent MTUs, it is possible that a probe will be delivered even if it is too large for some of the physical channels. In such cases raising the path MTU to the probe size can cause severe packet loss and abysmal performance. After raising the MTU, the new MTU size can be verified by monitoring the loss rate.

PLPMTUD introduces some flexibility in the implementation of classical path MTU discovery, which is subject to protocol failures (connection hangs) if ICMP PTB messages are not delivered or processed for some reason. With PLPMTUD, classical path MTU discovery can include additional consistency checks (e.g. validating additional fields in the transcribed header) without increasing the risk of connection hangs due to false failures of the added checks. Such changes to classical path MTU discovery are beyond the scope of this document.

In the limiting case, all ICMP PTB messages might be unconditionally ignored, and PLPMTUD can be used as the sole method used to discover the path MTU. In this configuration, PLPMTUD parallels congestion control. An end-to-end transport protocol adjusts non-protocol properties of the data stream (window size or packet size) while using packet losses to deduce the appropriateness of the adjustments.
This technique seems to be more philosophically consistent with the end-to-end principle of the Internet than relying on ICMP messages containing transcribed headers of multiple protocol layers.

Most of the difficulty in implementing PLPMTUD arises because it needs to be implemented in several different places within a single node. In general, each packetization protocol needs to have its own implementation of PLPMTUD. Furthermore, the natural mechanism to share path MTU information between concurrent or subsequent connections over the same path is a path information cache in the IP layer. The various packetization protocols need to have the means to access and update the shared cache in the IP layer. This memo describes PLPMTUD in terms of its primary subsystems without fully describing how they are assembled into a complete implementation.

Section 3 provides a complete glossary of terms.

Relatively few details of PLPMTUD affect interoperability with other standards or Internet protocols. These details are specified in RFC2119 standards language in section 4. The vast majority of the implementation details described in this document are recommendations based on experiences with earlier versions of path MTU discovery. These recommendations are motivated by a desire to maximize robustness of PLPMTUD in the presence of less than ideal network conditions as they exist in the field.

Section 5 describes how to partition PLPMTUD into layers, and how to manage the "path information cache" in the IP layer.

Section 6 describes the general Packetization Layer properties and features needed to implement PLPMTUD.

Section 7 recommends using IPv4 fragmentation in a configuration that mimics IPv6 functionality, to minimize future problems migrating to IPv6.

Section 8 describes the details of how to use probes to search for the path MTU.

Section 9 describes a programming interface for applications acting as packetization layers, and for tools to be able to diagnose path problems that interfere with path MTU discovery.

Section 10 discusses implementation details for specific protocols, including TCP.
3. Terminology

We use the following terms in this document:


Node: A device that implements IP.

Router: A node that forwards IP packets not explicitly addressed to itself.

Host: Any node that is not a router.

Upper layer: A protocol layer immediately above IP. Examples are transport protocols such as TCP and UDP, control protocols such as ICMP, routing protocols such as OSPF, and Internet or lower-layer protocols being "tunneled" over (i.e., encapsulated in) IP such as IPX, AppleTalk, or IP itself.

Link: A communication facility or medium over which nodes can communicate at the link layer, i.e., the layer immediately below IP. Examples are Ethernets (simple or bridged); PPP links; X.25, Frame Relay, or ATM networks; and Internet (or higher) layer "tunnels", such as tunnels over IPv4 or IPv6. Occasionally we use the slightly more general term "lower layer" for this concept.

Interface: A node’s attachment to a link.

Address: An IP-layer identifier for an interface or a set of interfaces.

Packet: An IP header plus payload.

MTU: Maximum Transmission Unit, the size in bytes of the largest IP packet, including the IP header and payload, that can be transmitted on a link or path. Note that this could more properly be called the IP MTU, to be consistent with how other standards organizations use the acronym MTU.

Link MTU: The Maximum Transmission Unit, i.e., maximum IP packet size in bytes, that can be conveyed in one piece over a link. Beware that this definition differs from the definition used by other standards organizations.

For IETF documents, link MTU is uniformly defined as the IP MTU over the link. This includes the IP header, but excludes link layer headers and other framing which is not part of IP or the IP payload.
Be aware that other standards organizations generally define link MTU to include the link layer headers.

Path: The set of links traversed by a packet between a source node and a destination node.

Path MTU, or pMTU: The minimum link MTU of all the links in a path between a source node and a destination node.

Classical path MTU discovery: Process described in RFC 1191 and RFC 1981, in which nodes rely on ICMP "Packet Too Big" (PTB) messages to learn the MTU of a path.

Packetization layer: The layer of the network stack which segments data into packets.

Effective PMTU: The current estimated value for PMTU used by a packetization layer for segmentation.

PLPMTUD: Packetization Layer Path MTU Discovery, the method described in this document, which is an extension to classical PMTU discovery.

PTB (Packet Too Big) message: An ICMP message reporting that an IP packet is too large to forward. This is the IPv6 term that corresponds to the IPv4 "ICMP Can’t fragment" message.

Flow: A context in which MTU discovery algorithms can be invoked. This is naturally an instance of the packetization protocol, e.g. one side of a TCP connection.

MSS: The TCP Maximum Segment Size [6], the maximum payload size available to the TCP layer. This is typically the path MTU minus the size of the IP and TCP headers.

Probe packet: A packet which is being used to test a path for a larger MTU.

Probe size: The size of a packet being used to probe for a larger MTU.

Probe gap: The payload data that will be lost and need to be retransmitted if the probe is not delivered.
Leading window: Any unacknowledged data in a flow at the time a probe is sent.

Trailing window: Any data in a flow sent after a probe, but before the probe is acknowledged.

Search strategy: The heuristics used to choose successive probe sizes to converge on the proper path MTU, as described in section Section 8.3.

Full stop timeout: a timeout where none of the packets transmitted after some event are acknowledged by the receiver, including any retransmissions. This is taken as an indication of some failure condition in the network, such as a routing change onto a link with a smaller MTU.

4. Requirements

All Internet nodes SHOULD implement PLPMTUD in order to discover and take advantage of the largest MTU supported along the Internet path.

Links MUST NOT deliver packets that are larger than their MTU. Links that have parametric limitations (e.g. MTU bounds due to limited clock stability) MUST include explicit mechanisms to consistently reject packets that might otherwise be nondeterministically delivered.

All hosts SHOULD use IPv4 fragmentation in a mode that mimics IPv6 functionality. All fragmentation SHOULD be done on the host, and all IPv4 packets, including fragments, SHOULD have the DF bit set such that they will not be fragmented (again) in the network. See Section 7.

The requirements below only apply to those implementations that include PLPMTUD.

To use PLPMTUD a Packetization Layer MUST have a loss reporting mechanism that provides the sender with timely and accurate indications of which packets were lost in the network.

Normal congestion control algorithms MUST remain in effect under all conditions except when only an isolated probe packet is detected as lost. In this case alone the normal congestion (window or data rate) reduction MAY be suppressed. If any other data loss is detected, standard congestion control MUST take place.

Suppressed congestion control (as above) MUST be rate limited such
that it occurs less frequently than the worst case loss rate for TCP congestion control at a comparable data rate over the same path (i.e. less than the "TCP-friendly" loss rate [@@]). This SHOULD be enforced by requiring a minimum headway between a suppressed congestion adjustment (due to a failed probe) and the next attempted probe, which is equal to one round trip time for each packet permitted by the congestion window. Alternatively this may be enforced by not suppressing congestion control if a 2nd probe is lost too soon after the 1st lost probe. This and other issues relating to congestion control are discussed in section @@window.

Whenever the MTU is raised, the congestion state variables MUST be rescaled so as not to raise the window size in bytes (or data rate in bytes per seconds).

Whenever the MTU is reduced (e.g. when processing ICMP PTB messages) the congestion state variable SHOULD be rescaled not to raise the window size in packets.

If PLPMTUD updates the MTU for a particular path, all Packetization Layer sessions that share the path representation SHOULD be notified to make use of the new MTU and make the required congestion adjustments.

All implementations MUST include a mechanism to implement diagnostic tools that do not rely on the operating systems implementation of path MTU discovery. This specifically requires the ability to send packets that are larger than the known MTU for the path, and collecting any resultant ICMP error message. See section 9 for further discussion of MTU diagnostics.

5. Layering

Packetization Layer Path MTU Discovery is most easily implemented by splitting its functions between layers. The IP layer is the best place to keep shared state, collect the ICMP messages, track IP header sizes and manage MTU information provided by the link layer interfaces. However, the procedures that PLPMTUD uses for probing, verification and scanning for the path MTU are very tightly coupled to features of the Packetization Layers such as data recovery and congestion control state machines.

Note that this layering approach is consistent with the advice in the current PMTUD specifications [2][3]. Many implementations of classical PMTU Discovery are already split along these same layers.
5.1. Accounting for header sizes

The way in which PLPMTUD operates across multiple layers requires a mechanism for accounting header sizes at all layers between IP and the packetization layer (inclusive). When transmitting non-probe packets, it is sufficient for the packetization layer to ensure an upper bound on final IP packet size, so as not to exceed the current effective path MTU. All packetization layers participating in classical path MTU discovery have this requirement already. When participating in PLPMTUD and transmitting a probe packet, the packetization layer must determine that packet’s final size including IP headers. This requirement is specific to PLPMTUD, and to satisfy it existing implementations may need additional inter-layer communication.

5.2. Storing PMTU information

This memo uses the concept of a "flow" to define the scope of the path MTU discovery algorithms. For many implementations, a flow would naturally correspond to an instance of each protocol, i.e., each connection or session. In such implementations the algorithms described in this document are performed within each session for each protocol. The observed PMTU can optionally be shared between different flows sharing a common path representation.

Alternatively, PLPMTUD could be implemented such that the complete PLPMTUD state is associated with the path representations. Such an implementation could use multiple connections or sessions for each probe sequence. This approach may converge much more quickly in some environments such as when an application uses many small connections, each of which may be too short to complete the path MTU discovery process.

These approaches are not mutually exclusive. However, due to differing constraints on generating probes (section Section 6.2) and the MTU searching algorithm (section @@@search), it may not be feasible for different packetization layer protocols to share PLPMTUD state. This suggests that it may be possible for some protocols to share probing state, but not others. In this case, the different protocols can still share the observed PMTU but they will have differing convergence properties.

The IP layer is the best place to store cached PMTU values and other shared state such as MTU values reported by ICMP PTB messages. Ideally this shared state should be associated with a specific path traversed by packets exchanged between the source and destination nodes. However, in most cases a node will not have enough information to completely and accurately identify such a path.
Rather, a node must associate a PMTU value with some local representation of a path. It is left to the implementation to select the local representation of a path.

An implementation could use the destination address as the local representation of a path. The PMTU value associated with a destination would be the minimum PMTU learned across the set of all paths in use to that destination. The set of paths in use to a particular destination is expected to be small, in many cases consisting of a single path. This approach will result in the use of optimally sized packets on a per-destination basis. This approach integrates nicely with the conceptual model of a host as described in [RFC 2461]: a PMTU value could be stored with the corresponding entry in the destination cache. Storing the minimum value is suggested since NATs and other forms of middle boxes may exhibit differing PMTUs at a single IP address.

Note that network or subnet numbers are not suitable to use as representations of a path, because there is not a general mechanism to determine the network mask at the remote host.

If IPv6 flows are in use, an implementation could use the IPv6 flow id [5][9] as the local representation of a path. Packets sent to a particular destination but belonging to different flows may use different paths, with the choice of path depending on the flow id. This approach will result in the use of optimally sized packets on a per-flow basis, providing finer granularity than MTU values maintained on a per-destination basis.

For source routed packets, i.e., packets containing an IPv6 routing header, or IPv4 LSRR or SSRR options, the source route may further qualify the local representation of a path. An implementation could use source route information in the local representation of a path.

5.3. Accounting for IPsec

This document does not take a stance on the placement of IPsec, which logically sits between IP and the Packetization Layer. The PLPMTUD implementation can treat IPsec either as part of IP or as part of the Packetization Layer, as long as the accounting is consistent within the implementation. If IPsec is treated as part of the IP layer, then each security association to a remote node may need to be treated as a separate path; i.e., the security association is used to represent the path. If IPsec is treated as part of the packetization layer, the IPsec header size must be included in the Packetization Layer’s header size calculations. [11]
5.4. Multicast

In the case of a multicast destination address, copies of a packet may traverse many different paths to reach many different nodes. The local representation of the "path" to a multicast destination must in fact represent a potentially large set of paths.

Minimally, an implementation could maintain a single MTU value to be used for all packets originated from the node. This MTU value would be the minimum MTU learned across the set of all paths in use by the node. This approach is likely to result in the use of smaller packets than is necessary for many paths.

If the application using multicast gets complete delivery reports (unlikely because this requirement has poor scaling properties), PLPMTUD could be implemented in multicast protocols.

6. Common Packetization Properties

This section describes general Packetization Layer properties and characteristics needed to implement PLPMTUD. It also describes some implementation issues that are common to all Packetization Layers.

6.1. Mechanism to detect loss

It is important that the Packetization Layer has a timely and robust mechanism for detecting and reporting losses. PLPMTUD makes MTU adjustments on the basis of detected losses. Any delays or inaccuracy in loss notification is likely to result in incorrect MTU decisions or slow convergence.

It is best if Packetization Protocols use fairly explicit loss notification such as selective acknowledgments, although implicit mechanisms such as TCP Reno style duplicate acknowledgments counting are sufficient. It is important that the mechanism can robustly distinguish between the isolated loss of just a probe and other combinations of losses.

Many protocol implementation have complicated mechanisms such as SACK scoreboards to distinguish between real losses and temporary missing data due to reordering in the network. In these implementations it is desirable to signal losses to PLPMTUD as a side effect of the data retransmission. This approach offers the maximum protection from confusing signals due to reordering and other events that might mimic losses.

PLPMTUD can also be implemented in protocols that rely on timeouts as
their primary mechanism for loss recovery; however, timeout should be used only when there are no other alternatives.

6.2. Generating probes

There are several possible ways to alter packetization layers to generate probes. The different techniques incur different overheads in three areas: difficulty in generating the probe packet (in terms of packetization layer implementation complexity and extra data motion) possible additional network capacity consumed by the probes and the overhead of recovering from failed probes (both network and protocol overheads).

Some protocols might be extended to allow arbitrary padding with dummy data. This greatly simplifies the implementation because the probing can be performed without participation from higher layers and if the probe fails, the missing data (the "probe gap") is assured to fit within the current MTU when it is retransmitted. This is probably the most appropriate method for protocols that support arbitrary length options or multiplexing within the protocol itself.

Many Packetization Layer protocols can carry pure control messages (without any data from higher protocol layers) which can be padded to arbitrary lengths. For example the SCTP HEARTBEAT message can be used in this manner (See section 10.2). This approach has the advantage that nothing needs to be retransmitted if the probe is lost.

These techniques do not work for TCP, because there is not a separate length field or other mechanism to differentiate between padding and real payload data. With TCP the only approach is to send additional payload data in an over-sized segment. There are at least two variants of this approach, discussed in section 10.1.

In a few cases there may no reasonable mechanisms to generate probes within the Packetization Layer protocol itself. As a last resort it may be possible to rely an an adjunct protocol, such as ICMP ECHO (aka "ping"), to send probe packets. See section 10.3 for further discussion of this approach.

7. Host Fragmentation

Packetization layers are encouraged to avoid sending messages that will require fragmentation. (For the case against fragmentation, see [14], [15]). However, entirely preventing fragmentation is not always possible. Some packetization layers, such as a UDP application outside the kernel, may be unable to change the size of
messages it sends, resulting in datagram sizes that exceed the path MTU.

IPv4 permitted such applications to send packets without the DF bit set. Oversized packets without the DF bit set would be fragmented in the network or sending host when they encountered a link with a MTU smaller than the packet. In some case, packets could be fragmented more than once if there were cascaded links with progressively smaller MTUs. This approach is not recommended.

It is recommended that IPv4 implementations use a strategy that mimics IPv6 functionality. When an application sends datagrams that are larger than the known path MTU they should be fragmented to the path MTU in the host IP layer even if they are smaller than the link MTU of the first network hop directly attached to the host. The DF bit should be set on the fragments, so they will not be fragmented again in the network.

This technique will minimize future surprises as the Internet migrates to IPv6. Otherwise, the potential exists for widely deployed applications or services relying on IPv4 fragmentation in a way that cannot be implemented in IPv6. At least one major operating system already uses this strategy.

The ability to selectively transmit packets larger than the current effective path MTU (but smaller than the link MTU) is required, to be able to send probes generated by packetization layers participating in PLPMTUD, and to facilitate diagnostic utilities.

Note that IP fragmentation divides data into packets, so it is minimally a Packetization Layer. However, it does not have a mechanism to detect lost packets, so it can not support a native implementation of PLPMTUD. Fragmentation-based PLPMTUD requires an adjunct protocol as described in section 10.3.

8. The Probing Method

This section describes the details of the MTU probing method, including how to send probes and process error indications necessary to search for the path MTU.

8.1. Packet size ranges

A packetization layer implementing PLPMTUD should keep three pieces of state:

Mathis & Heffner Expires April 26, 2006 [Page 17]
search_low: The smallest available probe size, minus one.
search_high: The greatest available probe size.
eff_pmtu: The effective pmtu for this flow.

<table>
<thead>
<tr>
<th>search_low</th>
<th>eff_pmtu</th>
<th>search_high</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

non-probe size range

<------------------------------------>

probe size range

Figure 1

When transmitting probes, the packetization layer will select the
probe size from within the range "(search_low, search_high]". When
transmitting non-probes, it may use sizes less than or equal to
eff_pmtu.

The eff_pmtu must be in the range "[search_low, search_high]". Note
that when probing upward eff_pmtu will equal search_low, but may
differ due to initial values, or ICMP PTB messages.

8.2. Selecting initial values

The initial value for search_high should be the largest possible
packet supported by the flow. This may be limited by the local
interface MTU, by a protocol mechanism such as the TCP MSS option, or
an intrinsic limit such as the protocol length field.

It is recommended that search_low be initially set to a value likely
to work over a large range of links. Given today’s technologies, a
value of 512 bytes is likely to work. For IPv6 flows, a value of
1280 is appropriate. The initial value for search_low should be
configurable.

Properly functioning Path MTU discovery is critical to the robust and
efficient operation of the Internet. Any major change (as described
in this document) has the potential to be very disruptive if it
contains any errors or oversights. The selection of initial values
determines to what extent a PLPMTUD implementation’s behavior differs
from classical PMTUD in cases where MTU discovery is not needed, or
where classical PMTUD is sufficient.

It may be desirable to configure hosts in such a way that PLPMTUD
only has an effect in cases where classical PMTUD fails. Setting
eff_pmtu equal to search_high and relying on black hole detection has
this effect. Using initial values of search_low = eff_pmtu =
search_high has the effect of effectively disabling PLPMTUD and
relying only on classical PMTUD.

In some cases where it is known that classical PMTUD is likely to fail, using a conservatively small initial eff_pmtu may produce better results. Using a small initial eff_pmtu may have no impact on protocol dynamics in all cases, but can result in much better performance by avoiding the costly timeouts required for black hole detection.

As appropriate initial values for PLPMTUD state variables may vary not only per host but per path, configuration options for these values in the route cache is desirable.

8.3. Selecting probe size

The probe may have a size anywhere in the "probe size range" described above. However, a number of factors affect the selection of an appropriate size. A simple strategy might be to do a binary search halving the probe size range with each probe. However, for some protocols, data in a lost probe may require retransmission, making a failed probe more expensive than a successful probe. For such protocols, a strategy using smaller probe sizes and "probing up" may behave better. For many protocols, both at and above the packetization layer, the benefit of increasing MTU sizes may follow a step function such that it is not advantageous to probe within certain regions at all.

As an optimization, it may be appropriate to probe at certain common or expected MTU sizes; for example, 1500 bytes for standard Ethernet, or 1500 bytes minus header sizes for tunnel protocols.

Some protocols may not even "choose" probe sizes. For protocols which have certain natural data block sizes, an effective strategy could be to simply treat blocks whose size falls in the probe size range as a probe.

Each packetization layer must determine when probing is considered converged; that is, when the probe size range is considered small enough that further probing is no longer worth its cost. When it is determined that searching has converged, a timer should be set for 5 minutes @why. When the timer expires, search_high should be reset to its initial value (described above) so that probing can resume. This is so that if the path changes, and increased path MTU is available, then the flow will eventually be able to take advantage of it to send larger packets.
8.4. Probing preconditions

Before sending a probe, the flow must at least meet the following conditions:

- The flow has no outstanding probes or losses.
- If the last probe failed or was inconclusive, then the probe timeout has expired (see Section 8.6.2).
- The available window is greater than the probe size.
- For a protocol which uses in-band data for probing, enough data is available to send the probe.

In order to allow loss detection mechanisms to be effective, some protocols may require a probe plus a number of non-probe packet’s worth of available data and window space.

When not enough data is available to probe, the protocol may wish to delay sending non-probes in order to accumulate enough data to send a probe.

8.5. Conducting a probe

Once a probe size in the appropriate range has been selected, and the above preconditions have been met, the packetization layer may conduct a probe. To do so, it creates a probe packet such that its size, including the outermost IP headers, is equal the probe size. After sending the probe it awaits response, which may take the following results:

- Success: The probe is acknowledged as having been received by the remote host.
- Failure: A protocol mechanism indicates that the probe was lost, but no packets in the leading or trailing window were lost.
- Timeout failure: A protocol mechanism indicates that the probe was lost, and no packets in the leading window were lost, but is unable to determine if any packets in the trailing window were lost. For example, loss is detected by a timeout, and go-back-n retransmission is used.
- Inconclusive: The probe was lost in addition to other packets in the leading or trailing windows.

8.6. Response to probe results

When a probe has completed, the result should be processed as follows, categorized by the probe’s result type.
8.6.1. Probe success

When the probe is delivered, this is an indication that the path MTU is at least as large as the probe size. The packetization layer should set search_low to the probe size, eff_pmtu to "max(eff_pmtu, probe size)".

Note that if a flow's packets are routed via multiple paths, or over a path with a non-deterministic MTU, delivery of a single probe packet does not indicate that all packets of that size will be delivered. To be robust in such a case, the packetization layer should conduct MTU verification as described in section @cite.

8.6.2. Probe failure

When only the probe is lost, this is treated as an indication that the path MTU is smaller than the probe size. In this case alone, the loss should not be interpreted as congestion signal.

In the absence of other indications, the packetization layer should set search_high to the probe size minus one, and eff_pmtu to "min(eff_pmtu, probe size)".

If an ICMP PTB message is received matching the probe packet, then search_high and eff_pmtu may be set from the MTU value indicated in the message. Note that the ICMP message may be received either before or after the protocol loss indication.

A probe failure event is the one situation under which the packetization layer is permitted not to treat loss as a congestion signal. Because there is some small risk that suppressing congestion control might have unanticipated consequences (even for one isolated loss), it is required that probe failure events be less frequent than the normal period for losses under standard congestion control. Specifically after a probe failure event and suppressed congestion control, PLPMTUD may not probe again until an interval which is comparable to the expected interval between congestion control events. This is required in section 4 and discussed further in section @window. The simplest estimate of the interval to the next congestion event is the same number of round trips as the current congestion window in packets.

8.6.3. Probe timeout failure

If the loss was detected with a timeout and repaired with go-back-n retransmission, then congestion window reduction will be necessary. The relatively high price of a failed probe in this case may merit a longer timeout. A timeout value of five times @@why? the non-timeout
failure case is recommended.

8.6.4. Probe inconclusive

The presence of other losses near the loss of the probe may indicate that the probe was lost due to congestion rather than because of an MTU limitation. In this case it is appropriate to update no state, and simply probe again when the probing preconditions are met; i.e., when no recent losses have been observed. At this point, it is particularly appropriate to re-probe since the flow’s congestion window will be at its lowest point, minimizing the probability of congestive losses.

8.7. Full stop timeout

Under all conditions a full stop timeout (also known as a "persistent timeout" in other documents) should be taken as an indication of some significantly disruptive event in the network, such as a router failure or a routing change to a path with a smaller MTU. For TCP, this occurs when the R1 timeout threshold described by [8] expires.

If there is a full stop timeout and there was not an ICMP message indicating a reason (PTB, Net unreachable, etc, or the ICMP messages was ignored for some reason), the suggested first recovery action is to treat this as a detected black hole as described in [10].

The response to a detected black hole should be to set search_low to its initial value, and set eff_pmtu to search_low. Upon further successive timeouts, search_low and eff_pmtu should be halved, with a lower bound of 68 bytes for IPv4 and 1280 bytes for IPv6.

8.8. MTU verification

It is possible for a flow to simultaneously traverse multiple paths, but it will only be able to keep a single path representation for the flow. If in such a case the paths have different MTUs, storing the minimum MTU of all paths in the flow’s path representation will result in correct, though sub-optimal behavior. If ICMP PTB messages are delivered, then classical PMTUD will work correctly in this situation.

However, if ICMP is not delivered, PLPMTUD will be relied upon and may fail because its requirement that links MUST NOT deliver packets larger than their MTU is effectively broken. A probe with a size greater than the minimum but smaller than the maximum of the path MTUs may be successful. However, upon raising the flow’s effective PMTU, the loss rate may significantly increase. The flow may still make progress, but the resultant loss rate may be unacceptable. For
example, when using two-way round-robin striping, 50% of full-sized packets would be lost.

Striping in this manner is often operationally undesirable (for example, due to packet reordering), and is usually avoided by hashing flows to a single path. However, to increase robustness an implementation should implement some form of MTU verification, such that if increasing eff_pmtu results in a sharp increase in loss rate, it will fall back to using a lower MTU.

A recommended strategy would be that when using a new value for eff_pmtu, to save the old value. If loss rate rises above a certain threshold for a period of time (for example, loss rate is higher than 10% over multiple RTO intervals), then the new MTU is considered incorrect. The saved value of eff_pmtu should be restored, and search_high reduced in the same manner as in a probe failure.

9. Diagnostic Interface

All implementations MUST include facilities for MTU discovery diagnostic tools that implement PPLPMTUD or other MTU discovery algorithms in user mode without help or interference by the PMTUD algorithm present in the operating system. This requires a mechanism where a diagnostic application can send packets that are larger than the operating system’s notion of the current path MTU and for the diagnostic application to collect any resulting ICMP PTB messages or other ICMP messages. For IPv4, the diagnostic application must be able to set the DF bit.

At this time nearly all operating systems support two modes for sending UDP datagrams: one which silently fragments packets that are too large, and another that rejects packets that are too large. Neither of these modes are suitable for efficiently diagnosing problems with MTU discovery, such as routers that return ICMP PTB messages containing incorrect size information.

10. Specific Packetization Layers

This section discusses specific implementation details for different protocols that can be used as Packetization Layer protocols. All Packetization Layer protocols must consider all of the issues discussed in section Section 6. For most protocols it is self evident how to address many of these issues. It is hoped that the protocols described here will be sufficient illustration for implementors to adapt other protocols.
10.1. Probing method using TCP

TCP has no mechanism that could be used to distinguish between real application data and some other form of padding that might be used to fill out probe packets. Therefore, TCP must generate probes by sending oversized segments that are carrying real data from upper layers. There are two approaches that TCP might use to minimize overhead associated with the probing sequence.

A TCP implementation of PLPMTUD can elect to send subsequent segments overlapping the probe as though the probe segment was not oversized. This has the advantage that TCP only needs to retransmit one segment at the current MTU to recover from failed probes. However the duplicate data in the probe does consume network resources and will cause duplicate acknowledgments. It is important that these extra duplicate acknowledgments not trigger Fast Retransmit. This can be guaranteed by limiting the largest probe segment size to twice the current segment size (causing at most 1 duplicate acknowledgment) or three times the current segment size (causing at most 2 duplicate acknowledgments).

The other approach is to send non-overlapping segments following the probe. Although this is cleaner from a protocol architecture standpoint it clashes with many of the optimizations used to improve the efficiency of data motion within many operating systems. In particular many implementations divide the data into segments and pre-compute checksums as the data is copied out of application buffers. In these implementations it can be relatively expensive to adjust segment boundaries after the data is already queued.

10.2. Probing method using SCTP

In the SCTP protocol [7][13] the application writes messages to SCTP and SCTP "chunkifies" them into smaller pieces suitable for transmission through the network. Once a message has been chunkified, they are assigned Transmission Sequence Numbers (TSNs). Once some TSNs have been transmitted SCTP can not change the chunk sizes. SCTP multi-path support normally requires SCTP to chunkify its messages to fit the smallest PMTU of all paths. Although not required, implementations may bundle multiple data chunks together to make larger IP packets to send on paths with a larger PMTU. Note that SCTP must independently probe the PMTU on each path to the peer.

The recommended method for generating probes is to add a chunk consisting only of padding to an SCTP message. There are two methods to implement this padding.

In method 1, the message is padded with an SCTP heart beat (HB), of
the necessary size to construct an IP packet the desired probe size. The peer SCTP implementation will acknowledge a successful probe without delay by returning the same Heartbeat as a HEARTBEAT-ACK. This method is fully compatible with current SCTP standards and implementations, but is exposed to MTU limitation on the return path, which might cause the HEARTBEAT-ACK to be lost.

In method 2, a new "PAD" chunk type would have to be defined. This chunk would be silently discarded by the peer. The PAD chunk could be attached to another message (either a minimum length HB or other application data which will be acknowledged by the peer) to build a probe packet. The default action for an unknown chunk type in the range 128 to 190, (high bits = 10 ) is to "Skip this chunk and continue processing" [RFC2960] - exactly the required behavior for a PAD chunk. Any currently unused type in this range will work for a PAD chunk type. This method is fully compatible with all current SCTP implementations, but requires adding a new type to the current standards. It has the advantage that restrictions due to the return path MTU are not applied to the forward path.

10.3. Probing method using IP fragmentation

As mentioned in section 7, datagram protocols (such as UDP) might rely on IP fragmentation as a packetization layer. However, implementing PLPMTUD with IP fragmentation is problematic because the IP layer has no mechanism to determine if the packets are ultimately delivered properly to the far node, without participation by the application.

To support IP fragmentation as a packetization layer under an unmodified application, we propose the use of an adjunct MTU measurement protocol (ICMP ECHO) and a separate path MTU discovery daemon (described here) to perform PLPMTUD and update the stored path MTU information.

For IP fragmentation the initial MTU should be selected as described in section 8.2, except with a separate global control for the default initial MTU for connectionless protocols. Since connectionless protocols may not keep enough state to effectively diagnose MTU black holes, it would be more robust to error on the side of using too small of an initial MTU (e.g. 1kBytes or less) prior to initiating probing of the path to measure the MTU.

Since many protocols that rely on IP fragmentation are connectionless, there is an additional problem with the path information cache: there are no events corresponding to connection establishment and tear-down to use to manage the cache itself. If there is no entry in the path information cache for a particular
packet being transmitted, it uses an immutable cache entry for the "default path", which has a MTU that is fixed at the initial value. A new path cache entry is not created until there is an attempt to set the MTU.

The path MTU discovery daemon should be triggered as a side effect of IP fragmentation. Once the number of fragmented datagrams via any particular path reaches some configurable threshold (say 5 datagrams), the daemon can start probing the path with ICMP ECHO packets. These probes must use the diagnostic interface described in section 9 and have DF set. The daemon can implement all of the PLPMTUD probe sequence and search strategy, collect all of the ICMP responses (ECHO REPLY, ICMP PTB, etc) and only the saved PTB in the path information cache in the IP layer.

Alternatively, most of the PLPMTUD state machinery can be implemented within the path information cache in the IP layer, which can specifically invoke the path MTU discovery daemon to perform specified measurements on specific paths and report the results back to the IP layer.

Using ICMP ECHO to measure the MTU has a number of potential robustness problems. Note that the most likely failures are due to losses unrelated to MTU (e.g. nodes that discriminate on the basis of protocol type). These non-MTU-related losses can prevent PLPMTUD from raising the MTU, forcing the packetization protocol to use a smaller MTU than necessary. Since these failures are not likely to cause interoperability problems they are relatively benign.

However there does exist other more serious failure modes, such as layer 3 or 4 routers choosing different paths for different protocol types or sessions. In such environments, adjunct protocols may experience different MTUs than the primary protocol. If the adjunct protocol has a larger MTU than the primary protocol, PLPMTUD will select a non-functional MTU. This does not seem to be a likely situation.

10.4. Probing method using applications

The disadvantages of probing with ICMP ECHO can be overcome by implementing the path MTU discovery daemon within the application itself, using the application’s own protocol.

The application must have some suitable method for generating probes. The ideal situation is a lightweight echo function, that confirms message delivery, plus a mechanism for padding the messages out to the desired MTU, such that the padding is not echoed. This combination (akin to the SCTP HB plus PAD) is preferred because you
can send large probes that cause small acknowledgments. For protocols that cannot implement these messages directly, there are often alternate methods for generating probes. For example, the protocol may have a variable length echo (that measures both the forward and return path) or if there is no echo function, there may be a way to add padding to regular messages carrying real application data. There may also be other ways to generate probes. As a last resort, it may be feasible to extend the protocol with new message types to support MTU discovery.

Probing within an application introduces one new issue: many applications do not currently concern themselves with MTU and rely on IP fragmentation to deliver datagrams that just happen to be larger than the path MTU. PLPMTUD requires that the protocol be able to send probes that are larger than the IP layer’s current notion of the path MTU, but are marked not to be fragmented. This requires an alternate method for sending these datagrams.

As with ICMP MTU probing, there is considerable flexibility in how the PLPMTUD algorithms can be divided between the Application and the path information cache.

Some applications send large datagrams no matter what the link size, and rely on IP fragmentation to deliver the datagrams. It has been known for a long time that this has some undesirable consequences [@harm1]. Recently it has come to light that IPv4 fragmentation is not sufficiently robust for general use in today’s Internet. The 16-bit IP identification field is not large enough to prevent frequent misassociated IP fragments and the TCP and UDP checksums are insufficient to prevent the resulting corrupted data from being delivered to higher protocol layers. [@harm2]

None the less, there are a number of higher layer protocols, such as NFS [@NFS] which use IP fragmentation as a mechanism to reduce CPU load. NFS typically sends fragmented 8k Byte datagram’s over all link types, no matter what the link MTU. The other common case, in which the application wants to use the largest possible datagram that fits within the MTU is most easily treated as a special case of the fragmenting case.

11. References
11.1. Normative references


November 1990.


11.2. Informative references


Appendix A. Security Considerations

Under all conditions the PLPMTUD procedure described in this document is at least as secure as the current standard path MTU discovery procedures described in RFC 1191 [2] and RFC 1981 [3].

Since this algorithm is designed for robust operation without any ICMP (or other messages from the network), PLPMTUD could be configured to ignore all ICMP messages (globally or on a per application basis). In this configuration, it cannot be attacked unless the attacker can identify and selectively cause probe packets to be lost.

Appendix B. IANA Considerations

None.

Appendix C. Acknowledgements

Many ideas and even some of the text come directly from RFC1191 and RFC1981.

Many people made significant contributions to this document, including: Randall Stewart for SCTP text, Michael Richardson for material from an earlier ID on tunnels that ignore DF, Stanislav Shalunov for the idea that pure PLPMTUD parallels congestion control, and Matt Zekauskas for maintaining focus during the meetings. Thanks to the early implementors: Kevin Lahey, John Heffner and Rao Shoaib who provided concrete feedback on weaknesses in earlier drafts. Thanks also to all of the people who made constructive comments in the working group meetings and on the mailing list. I am sure I have missed many deserving people.

Matt Mathis and John Heffner are supported in this work by a grant from Cisco Systems, Inc.
Authors’ Addresses

Matt Mathis
Pittsburgh Supercomputing Center
4400 Fifth Avenue
Pittsburgh, PA  15213
US

Phone: 412-268-3319
Email: mathis@psc.edu

John W. Heffner
Pittsburgh Supercomputing Center
4400 Fifth Avenue
Pittsburgh, PA  15213
US

Phone: 412-268-2329
Email: jheffner@psc.edu
Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Copyright Statement

Copyright (C) The Internet Society (2005). This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.