Network Working Group
Internet-Draft
Expires: August 21, 2005

M. Mathis
J. Heffner
PSC
K. Lahey
Freelance

February 20, 2005

Path MTU Discovery
draft-ietf-pmtud-method-04

Status of this Memo

Copyright Notice

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

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 draws heavily RFC 1191 [2] and RFC 1981 [3] for terminology, ideas and some of the text.

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 -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. Passes 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 relying on explicit signals from the network. 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 acknowledgement structure that delivers to the sender accurate and timely indications of which packets were lost.

The general strategy of the new procedure is for the packetization layer to find an appropriate path MTU by probing with progressively larger packets. A "probe sequence" consists of a single "probe packet", which initiates a "probe phase", followed by a "transition phase" and a "verification phase".

If a probe packet is successfully delivered, then the path MTU is provisionally raised to the probe size during the transition phase. If there are no losses during the subsequent verification phase, then the path MTU is confirmed (verified) to be at least as large as the provisional MTU. Each conclusive probe sequence narrows the MTU search range, converging toward the true path MTU.

The verification phase is 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 will cause severe periodic loss and abysmal performance. The verification phase is designed to prevent the path MTU from being raised if doing so causes excessive packet losses.

A conservative implementation of PLPMTUD would use a full round trip time for the verification phase. In this case the entire probe sequence takes three full round trip times. It takes one round trip for the probe phase, during which the probe propagates to the far node and an acknowledgment is returned. The second round trip is the transitional phase, during which data packets using the provisional MTU propagate to the far node and are acknowledged. During he third and final round trip time, it is verified that raising the MTU did not cause any additional losses.

The isolated loss of a probe packet (with or without an ICMP PTB 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 any of the three phases, the loss is treated as a congestion indication as well as an indication of some sort of failure of the PLPMTUD process. The congestion indication is treated like any other congestion indication: window or rate adjustments are mandatory per the relevant congestion control standards [8]. Probing can resume after a delay which is determined by the nature of the detected failure.

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.

There are a range of options for integrating PLPMTUD with classical path MTU discovery. In the most conservative configuration, from a deployment point of view, classical path MTU discovery is fully functional (all correct ICMP PTB messages are unconditionally processed) and PLPMTUD is invoked only to recover from ICMP black holes.

In the most conservative configuration, from a security point of view, all ICMP PTB messages are ignored, and PLPMTUD is the sole method used to discover the path MTU. This protects against malicious or erroneous ICMP PTB messages which might otherwise cause MTU discovery to arrive at the incorrect MTU for a path.

Note that in the latter 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.

We advocate a compromise, in which ICMP PTB messages are only processed in conjunction with probing (described in section 6.2.1), and Packetization Layer timeouts (described in section 6.2.3), and ignored in all other situations.

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 details of a probe sequence, including how to process MTU and error indications, necessary to raise the MTU by one step.

Section 7 describes the general search strategy and Packetization Layer features needed to implement PLPMTUD.

Section 8 discusses specific implementation details for some specific protocols, including TCP.

Section 9 describes ways to minimize deployment problems for PLPMTUD, by including a number of good management features. It also addresses some potentially serious interactions with nodes that do not honor the IPv4 DF bit.

3. Terminology

We use the following terms in this document:

IP: Either IPv4 [1] or IPv6 [7].

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 differers 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.

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.

MPS: The maximum IP payload size available over a specific path. Typically this is the path MTU minus the IP header. As an example, this is the maximum TCP packet size, including TCP payload and headers but not including IP headers. This has also been called the "Layer 3 MTU".

MSS: The TCP Maximum Segment Size, the maximum payload size available to the TCP layer. This is typically the path MPS minus the size of the TCP header.

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.

Successful probe: The probe packet was delivered through the network and acknowledged by the Packetization Layer on the far node.

Inconclusive probe: The probe packet was not delivered, but there were other lost packets close enough to the probe where it can not be presumed that the probe was lost because it was larger than the path MTU. By implication the probe might have been lost due to something other than MTU (such as congestion), so the results are inconclusive.

Failed probe: The probe packet was not delivered and there were no other lost packets close to the probe. This is taken as an indication that the probe was larger than the path MTU, and future probes should be smaller.

Errored probe: There were losses or timeouts during the verification phase which suggest a potentially disruptive failure or network condition. These are generally retried only after substantially longer intervals.

Probe gap: The payload data that will be lost and need to be retransmitted if the probe is not delivered.

Probe phase: The interval (time or protocol events) between when a probe is sent and when it is determined that the the probe succeeded, failed or was inconclusive

Verification phase: An additional interval during which the new path MTU is considered provisional. Packet losses or timeouts are treated as an indication that there may be a problem with the provisional MTU.

Transition phase: The interval between the probe phase and the verification phase, during which packets using the new MTU propagate to the far node and the acknowledgment propagates back.

Probe sequence: The sequence of events to raise the MTU by one step, starting with the transmission of a probe packet followed by probe, transition and verification phases.

Search strategy: The heuristics used to choose successive probe sizes to converge on the proper path MTU, as described in section 7.5.

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. For the sake of PLPMTUD we suggest the following definition of a full stop timeout: the loss of one full window of data and at least one retransmission or at least 6 consecutive packets including at least 2 retransmissions (along with two retransmission timer expirations). [@@@ This probably needs some experimentation.]

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 6.4.

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 7.6.

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 9.4 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 the data recovery and congestion control state machines in the Packetization Layers. The most difficult part of implementing PLPMTUD is properly splitting the implementation between the layers.

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

Early implementation of PLPMTUD revealed that it is critically important to have a good clean mechanism for accounting header sizes at all layers. This is because each Packetization Layer does its calculations in its own natural data unit, which are almost always a reflection of the service that the Packetization Layer provides to the application or other upper layers. For example, TCP naturally performs all of its calculations in terms of sequence numbers and segment sizes. However, the MTU size being probed, MTU size reported in ICMP PTB messages, etc are measures of full packets, which not only include the TCP payload (measured in sequence space) but also include fixed TCP and IP headers, and may include IPv6 extension headers or IPv4 options, TCP options and even IPsec AH or ESP headers.

PLPMTUD requires frequent translation between these two domains: the Packetization Layer's natural data unit and full IP packet sizes. While there are a number of possible ways to accurately implement dual size measures, our experience has been that it is best if the boundary between the IP layer and the Packetization layer communicate

in terms of the IP Maximum Payload Size or MPS. The MPS is the only size measure that is common to both layers because it exactly matches the boundary between the layers. The IP Layer is responsible for adding or deducting its own headers when translating between MTU and MPS. Likewise the Packetization Layer is responsible for adding or deducting its own headers when calculations in its natural data units. For example, the MPS and TCP's MSS are different by the TCP header size.

Be aware that a casual reading of this document might give the impression that MTU, MPS and Packetization Layer data size (e.g. TCP MSS) are used interchangeably. They are not. Our choice of terminology is consistent with the protocol layer being discussed in the surrounding context. All implementors must pay attention to the distinction between these terms and include all necessary conversions, even when thy are not explicitly indicated in this document.

5.2 Storing PMTU information

The IP layer is the best place to store cached MPS 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 MPS 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 MPS value associated with a destination would be the minimum MPS 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 MPS value could be stored with the corresponding entry in the destination cache. However, NAT and other forms of middle boxes may exhibit differing MTUs at as 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 [7][14] 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 MPS 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. As far as PLPMTUD is concerned IPsec can be treated 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 the security association is used to represent the path. If IPsec is treated as part of the packetization layer, the IPsec header size has to be included in the Packetization Layer's header size calculations.

5.4 Measuring path MTU

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 MPS can 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. For example, one connection could do the initial probe and set the provisional MTU and one or more subsequent connection could verify the MTU. This approach may converge much more quickly in some environments such as when the application uses many small connections, each of which is too short to complete a probe sequence.

These approaches are not mutually exclusive. However, due to differing constraints on generating probes (section Section 7.2) and the MPS searching algorithm (section Section 7.5), 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 MPS but they will have differing convergence properties.

6. The Probing Sequence and Lower Layers

This section describes the details of a probe sequence, including how to process MTU and error indications, necessary to raise the MTU by one step.

6.1 Normal sequence of events to raise the MTU

If the probe size is smaller than the actual path MTU and there are no other losses, the normal sequence of events to raise the MTU is:

1. Confirm probing preconditions: no outstanding Packetization Layer losses, sufficient congestion window per section 7.6, sufficient elapsed time since previous probe per section 6.3, if candidate MPS has not been set from ICMP MPS, then compute the candidate MPS per MPS search strategy in section 7.5.

2. A new MTU is tested by sending one "probe packet", of size "probe size" (computed from the candidate MPS). The probe is sent, followed by additional packets at the current MTU. By definition PLPMTUD enters the probe phase. The probe propagates through the network and the far node acknowledges it (or possibly latter data, if acknowledgments are cumulative and delayed acknowledgment is in effect).

3. The acknowledgment for the probe reaches the data sender. By definition, this ends the probe phase.

4. The packetization layer provisionally raises the MTU to the probe size. PLPMTUD enters the transitional phase when it starts sending data using the provisional MTU.

   Note that implementations that use packet counts for congestion accounting (e.g. keep cwnd in units of packets) must re-scale their congestion accounting such that raising the MTU does not raise the data rate (bytes/second) or the total congestion window in bytes, as required in section 4 and discussed in 7.6.

   If the implementation packetizes the data at the application programming interface, it may transmit already queued data at the current MTU before raising the MTU. In this case this data is not part of either the probing or transition phases, because all of the packets in flight fit within the current MTU.

5. Once the first packet of the transitional phase is acknowledged, PLPMTUD enters the verification phase. In principle the verification phase can be of arbitrary duration, however at this time we are recommending one full window of data (i.e one full round trip time) for most Packetization Layers.

6. Once there has been sufficient data delivered and acknowledged the provisional MTU is considered verified and the path MTU is updated. PLPMTUD can then probe for an even larger MTU, as described in the searching strategy in section 7.5.

Other events described in the next section are treated as exceptions and alter or cancel some of the steps above.

6.2 Processing MTU Indications

When the probe sequence fails to raise the MTU, it will be due to one of three broad classes of outcomes: the probe was inconclusive, failed or errored. If the probe was inconclusive, it means that there were other losses seemingly unrelated to the probe, such that the probe outcome was ambiguous. Inclusive probes should be retried with the same probe size. If the probe failed, this is an indication that the probe size was larger than the path MTU, and probing should continue with a smaller size, as selected by the MTU searching algorithm. In some situations there can be indications that the probing sequence caused some unexpected event. In these error conditions, it is desirable to use progressively longer delays between probes to minimize the possible impact on the network.

6.2.1 Processing ICMP PTB messages

Classical PMTU discovery specifies the generation of ICMP PTB Messages if an over-sized packet (e.g. a probe) encounters a link that has a smaller MTU. Since these messages can not be authenticated they introduce a number of well documented attacks against classical PMTUD [5].

With PLPMTUD these messages are not required for correct operation, and in principle can be summarily ignored at the expense of slower convergence to the proper MTU. However, we believe that a slightly better approach is to save the reported PTB size (computed from the ICMP MTU) in the path information cache and act on it only in conjunction with a lost PLPMTUD probe or a full-stop timeout.

Every ICMP PTB Message should be subjected to the following checks:
o If globally forbidden then discard the message.

- If forbidden by the application then discard the message.

- If this path has been tagged "bogus ICMP messages" then discard the message.

- If the reported MTU fails consistency checks then set "bogus ICMP messages" flag for this path and discards the message. These consistency checks include:
  * unrecognized or unparseable enclosed header, or
  * reported MTU is larger than the size indicated by the enclosed header, or
  * larger than the current MTU, provisional MTU or probe size as appropriate, or
  * fails a ICMP consistency checks specific to the Packetization Layer. (E.g. The SCTP Verification-Tag mechanism [9][16])

  To ease migration, it is suggested that implementations may include global controls to emulate legacy operation by suppressing some or all of the consistency checks.

If the ICMP PTB message is acceptable under all of these checks then save the "ICMP MPS" computed from the MTU field in the ICMP message. If the global configuration switch is set to emulate classical path MTU discovery then process the message immediately, i.e., set the path MPS to the ICMP MPS and invoke any protocol specific actions. Otherwise, the saved ICMP MPS will be acted upon if and only if there are other PLPMTUD events such as lost probes, etc as indicated in the next section. This delayed processing of ICMP PTB messages makes it more difficult for an attacker to interfere with correct PLPMTUD operation by injecting fraudulent ICMP PTB messages.

In either case if the Packetization Layer calls for specific actions in response to a PTB message, that action should be invoked only at the point when the path MPS is updated from the ICMP MPS.

6.2.2 Packetization Layer Detects Lost Packets

Each packetization protocol has its own mechanism to detect lost packets and request the retransmission of missing data. The primary signals used by PLPMTUD are these protocol-specific loss indications. The packetization layer is responsible for retransmitting the lost data if necessary, and notifying PLPMTUD that there was a loss.

- If the probe itself was lost, and there were no other losses during the probe phase (The RTT between when the probe was sent and the loss detected) then it is taken as an indication that the path MTU is smaller than the probe size. In this specific situation, the Packetization Layer may choose not to treat this

loss as a congestion signal, and continue with the same congestion window or data transmission rate.

If an accepted ICMP PTB message was received after the probe was sent, and it passes the additional checks that the ICMP MTU value is less than the probe size, and corresponds to an MPS greater than that in use for the path, then set the candidate MPS from the ICMP MTU value, and restart the probe sequence from step 1 in section 6.1.

If there was not an accepted PTB Message, then the indicated event is a "probe failure", which can be retried with a smaller probe size after a suitable delay for a probe_fail_event. See section 6.2.2 for more complete descriptions of failure events.

- If there are losses during the probe phase yet the probe was acknowledged as received, then the probe was successful. However, since additional losses have the potential to spoil the verification phase, it is important that PLPMTUD not progress into the transition phase (step 4 above) until after the Packetization Layer has fully recovered from the losses and completed the congestion window (or rate) adjustment.

- If there are losses during the probe phase and the probe was also lost the outcome depends on the presence an ICMP MTU set by an acceptable PTB message.

  If there was an accepted PTB message received after the probe was sent, it should be treated in the same manner as if there were no other losses (see above).

  If there was not an acceptable ICMP PTB message, then the probe is inconclusive because the lost probe might have been caused by congestion. The probe can be retried after a suitable delay for a probe_inconclusive_event.

- It is unlikely that losses during the transition phase are caused by PLPMTUD; however, the presence of loss does potentially complicate the verification phase. Note that we are referring to losses that are bracketed by acknowledgment of packets that were sent at the old MTU, while the transition to the provisional MTU is still propagating through the network. The first acknowledgment from the provisional MTU (and the transition to the verification phase) is most likely going to occur during the recovery of the losses in transition phase. It is important that the Packetization Layer retransmission machinery distinguish between losses at the old MTU (transition phase) and the provisional MTU (the verification phase, discussed next).

- Losses during the verification phase are taken as an indication that the path may have a non-uniform MTU or other condition such that raising the MTU raises the loss rate. If so, this is potentially a very serious problem. The provisional MTU is considered unsuitable, and the cached path MTU is set back to the previously verified MTU.

  Packet loss during the verification phase might also be due to coincidental congestion on the path, unrelated to the probe, so it would seem desirable to re-probe the path. The risk is that this effectively raises the tolerated loss threshold because even though raising the MTU seemed to cause additional loss, there is a statistical chance that repeated attempts to verify a new MTU may yield as false pass. The compromise is to re-probe once with the same probe size (after delay probe_inconclusive_event), and if this also fails, then the probe may not be retried until after a suitable delay for a verification_error_event, which exponentially increases on each successive failure.

6.2.3 Packetization Layer Retransmission Timeout

Note that the we do not make distinctions between the various methods that different Packetization Layers might use for detecting and retransmitting lost packets. It is preferable that the Packetization Layer uses a recovery mechanism similar to TCP SACK or fast retransmit designed to detect and report losses to recover as quickly as possible.

Under some conditions the Packetization Layer may have to rely on retransmission timeouts or other fairly disruptive techniques to detect and recover from losses. Since these greatly increase the cost of failed probes, it is recommended that PLPMTUD use even longer delays before re-probing. In these situations replace probe_fail_event with probe_timeout_event.

6.2.4 Packetization Layer Full Stop Timeout

Under all conditions (not just during MTU probing) a full stop timeout 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.

If an ICMP PTB message was recently received, even if its its MTU value was less than the current path MTU value in use, then the path MTU can be reduced to the ICMP MTU. A full stop timeout is the only situation outside of a probe that we recommended that the path MTU is set from the ICMP MTU. (In section 9.1 we relax this recommendation to facilitate migration to PLPMTUD in exchange for slightly less protection from corrupt ICMP PTB messages).

Note that whenever a problem with the path that causes a full-stop timeout (also known as a "persistent timeout" in other documents), several different path restart/recovery algorithms may be invoked at different layers in the stack. Some device drivers may be restarted [@@], router discovery [@@], ES-IS [@@] and so forth. We recommend that in most situation the first action should be to reset the path MTU down. Note that this recommendation is really beyond the scope of this document, and may require substantial additional research.

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), we suggest that the first recovery action should be to set the path MTU down to a safe minimum "restart MTU" value, and the reset PLPMTUD search state, so PLPMTUD will start over again searching for the proper MTU. The default IPV4 restart_MTU should be the minimum MTU as specified by IPv4 (576 Bytes)[1]. The default IPV6 restart_MTU should be the minimum MTU as specified by IPv6 (1280 Bytes) [7]. Unless the default MTU is overridden by some global control (See section 9.5).

If, and only if, the full stop timeout happens during the probe or transition phases, e.g., after sending data using the provisional MTU but before any of it is acknowledged, is it considered likely that raising the MTU caused the full stop timeout. If so, this situation is is likely to be cyclic, because resetting the PLPMTUD search state is likely to eventually cause re-probing the same problematic MTU. It is tempting to define additional states to detect recurrent full stop timeouts. However in today's hostile network environment, there is little tolerance for nodes that are so fragile that they can be disrupted by something as simple as oversized packets. Therefore, we do not feel that it is worth the overhead of specifying a state machine that is capable of automatically detecting these situations and disabling PLPMTUD. However, it is important that there be a manual way to disable or limit probing on specific paths. See section 9.5.

6.3 Probing Intervals

The previous sections describe a number of events that prevent a probe sequences from raising the path MTU. In all cases the basic response is the same: to wait some time interval (dependent on the specific event and possibly the history) and then to probe again. For events that are "inconclusive," it is generally appropriate to re-probe with the same probe size. For events that are identified as "failed probes," it is generally appropriate to re-probe with a smaller probe size. The search strategy described in section 7.5 is used to select probe sizes.

Many of the intervals described below are specified in terms of elapsed round trips relative to the current congestion window. This is because TCP and other Packetization Layer protocols tend to exhibit periodic loses which cause periodic variations of the congestion window and possibly the data rate. It is preferable that the PLPMTUD probes be scheduled near the low point of these cycles to minimize ambiguities caused by congestion losses.

In order from least to most serious:

probe_converge_event: The candidate probe size has already been probed so there is no need for further searching. Delay 5 minutes and then re-probe last SEARCH_HIGH.

probe_inconclusive_event: Other lost packets near the lost probe made the probe result ambiguous. Since the loss of non-probe packets requires a window (or data rate) reduction, it is desirable to schedule the re-probe (at the same probe size) roughly one round trip time after the end of the loss recovery. This will be almost the minimum congestion window size, with a small cushion to minimize the chances that correlated losses caused by some other bursty connection spoil another probe.

probe_fail_event: A probe fail 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), we require that probe fail events be less frequent than the normal period for losses under standard congestion control. Specifically after a probe fail 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 7.6.

The simplest estimate of the interval to the next congestion event is the same number of round trips as the current window in packets.

probe_timeout_event: Since this event was detected by a timeout, it is relatively disruptive to protocol operation. Furthermore, since the event indirectly includes a window adjustment that may have been caused by the MTU probe, it is important that the probe not be repeated until congestion control has had more than sufficient time to recover from the loss. Therefore we recommend five times the probe_fail_event interval, i.e., five times as many round trips as the current congestion window in packets.

verification_error_event: A verification fail event indicates that a probe was delivered and the verification phase failed twice separated by a congestion adjustment (so the second verification phase was at a low point in the congestion control cycle). This is an indication that one of the following three things might have happened: repeated losses unrelated to PLPMTUD; the path is striped across links with dissimilar MTUs, or the link layer has some parametric limitation such that raising the MTU greatly increases the random error rate.

The optimal method responding to this situation is an open research question. We believe that the correct response is some combination of exponentially lengthening back-offs, e.g., starting at 1 minute and quadrupling on each repeat, and implicitly treating the situation as a probe fail (and choosing a smaller probe size) after some threshold number of repeated verification_error_events.

6.4 Host fragmentation

Packetization layers are encouraged to avoid sending messages that will require fragmentation. (For the case against fragmentation, see [17], [18]). 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 no longer recommended. We now recommend 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.

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 8.3.

6.5 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 MPS value to be used for all packets originated from the node. This MPS value would be the minimum MPS 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.

7. 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.

7.1 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 implementation is desirable to signal losses to PLPMTUD as a side effect of the data

retransmission. This approach offer 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, although this should be used only when there are no other alternatives.

7.2 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 it this manner (See section 8.2) . 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 8.1.

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 8.3 for further discussion of this approach.

7.3 Mechanism to support provisional MTUs

The verification phase requires a mechanism provisionally raise the

MPS and if there are additional losses, restore the old MPS. While this is not difficult for most potential Packetization Layers, there are a few (e.g. ISO TP4 [ISOTP]) that are not allowed to re-packetize when doing a retransmission. That is, once an attempt is made to transmit a segment of a certain size, the transport cannot split the contents of the segment into smaller segments for retransmission. In such a case, the original segment can be fragmented by the IP layer during retransmission as described in section 6.4. Subsequent segments, when transmitted for the first time, should be no larger than allowed by the path MTU.

Note that while padding is an appropriate mechanism for probing, it is too wasteful for use during the verification phase.

Unresolved problem: if 2 PL are using the same path and one can only verify constrained sizes (e.g blocks+headers) then the verified MTU might be the actual packet size for the constrained PL, not the probed size. @@@@

Unresolved problem: what to do about very short flows? No verification phase? @@@@@

7.4 Selecting the initial MPS

If if there is already a cached MPS value for this path, PLPMTUD may use the saved MPS value. Unless it is very recent (how recent? @@@@@) SEARCH_HIGH should be set to SEARCH_MAX, to restart the search process from the old MPS.

Note that there are tradeoffs to how long the path information cache entries is retained when it is not being used by any flows. If they are kept for to long they waste memory, if too short it will cause frequent re-probing. We suggest an adjustable Least Recently Used algorithm to purge old entries. @@@@ This belongs some place else.

When the PLPMTUD process is started the recommended initial MPS should normally be set such that the Packetization Layer can carry 1 kByte data segments. This initial MPS would be 1 kByte plus space for Packetization layer headers. (see section 5 on accounting for headers). With the this MPS, RFC2414 [6] allows TCP and other transport protocols to start with an initial window of 4 packets.

[We suspect, but have not confirmed that] TCP completes sooner for short connections when started with four 1kB packets rather than three 1500 byte packets because the 2nd ACK occurs one round trip earlier

This initial MPS should also be configurable. One of the

configuration options should be to mimic classical PMTUD behavior by setting the initial MPS from the interface MTU. This option facilitates using PLPMTUD in a mode that mimics classical PMTU discovery. (See section 9.1)

7.5 Common MPS Search Strategy

The MPS search strategy described here is a only rough guide for implementors. It is difficult to imagine a completely standard algorithm because the strategy can include many Packetization Layer specific heuristics to optimize MPS selection. There is significant opportunity for future improvements to this portion of PLPMTUD.

The search strategy is trying to find the largest "candidate MPS" that meets the constraints of both the Packetization and the link layers. Although this algorithm is primarily described in terms of MPS, it needs to use knowledge about link layer MTUs and Packetization Layer buffer sizes.

The search strategy uses three variables:

SEARCH_MAX is the largest MPS that a Packetization Layer might be able to use. It is determined by such considerations as interface MTU, widths of protocol length fields, and possibly other protocol-dependent values, such as the the TCP MSS option. In many cases it would be the same as the classical MTU discovery initial MTU, minus the IP layer headers.

SEARCH_LOW is the largest validated MPS, the same as them current MPS in use by the packetization layer. The initial value for SEARCH_LOW is described in section 7.4.

SEARCH_HIGH is the least invalidated MPS. In most cases is will be the most recent failed candidate MPS. When PLPMTUD is initialized SEARCH_HIGH should be set to SEARCH_MAX, indicating that there have been no failed probes.

For many Packetization Layer protocols, the cost for a failed probe is significantly higher than the cost of a successful probe due to the additional time and overhead needed for retransmission and recovery. For this reason it is often desirable to bias the search strategy to make more smaller steps.

The search strategy first computes an initial candidate MPS using one of these methods:

If SEARCH_HIGH >= SEARCH_MAX, there have been no recent failed probes so use a coarse (geometric doubling) scan. Set candidate MPS = MIN(2 * SEARCH_LOW, SEARCH_MAX). Otherwise use one of several possible fins scan candidate MPS values:

Select a candidate MPS that corresponds to a common MTU possibly minus common tunnel header sizes between SEARCH_LOW and

SEARCH_HIGH. There is a fine scan heuristic described section 7.5.1 that might be used.
Use a simple weighted binary search by selecting the candidate MPS some prorated distance between SEARCH_LOW and SEARCH_HIGH. E.g. set
candidate MPS = SEARCH_LOW * (1 - alpha) + SEARCH_HIGH * alpha, for some alpha between 0 and 1. If you choose an alpha slightly less than 0.5, PLPMTUD will tend to converge from below, minimizing the number of failed probes. Alternatively alpha can be selected to optimally converge for some common MTUs, such as 1500 bytes.

If the Packetization Layer has preferred data sizes (e.g. carries block data), optionally round the candidate MPS to an efficient size for the Packetization Layer. The rounded candidate MPS would typically be a multiple of the optimal data block size plus space for Packetization Layer headers. The MPS can be rounded up or down, but should avoid selecting previously probed valued if possible, per the convergence test below. Packetization Layer that do not have intrinsically preferred data sizes may still choose to round the candidate MPS to some convenient increment such as 4 or 8 bytes, to prevent excessive hunting. Note that this step is intrinsically Packetization Layer dependent, and may be different for different packetization Layers.

If the resulting candidate MPS is not between SEARCH_LOW and SEARCH_HIGH, then the probe process has converged and further probing will not yield a better value for the MPS for this protocol. To detect if a routing change has raised the path MTU, the path should be re-probed after a suitable delay as indicated by a probe_converge_event (See section 6.3). If the probe succeeds, then SEARCH_HIGH should be set to SEARCH_MAX to restart the probing process from the current MPS.

MPS searching can be implicitly disabled by setting the SEARCH_HIGH to SEARCH_LOW.

Note that if two different Packetization Layers are sharing a path, they may choose different MPS due to differences in the protocols. It is even possible for one of the Packetization Protocol to consider the process converged, while the other continues to probe. In this case one of the Packetization Layers does may chose not to use the full MPS, and instead chooses some slightly smaller but more efficient packet size.

7.5.1 Fine Scans

If SEARCH_LOW does not correspond to a common link MTU, and there is a common link MTU between SEARCH_LOW and SEARCH_HIGH, set the

candidate MPS from the most common link MTU between SEARCH_LOW and SEARCH_HIGH.

If SEARCH_LOW does not correspond to a common link MTU, and there is not a common link MTU between SEARCH_LOW and SEARCH_HIGH, then set the candidate MPS to either the weighted binary search between SEARCH_LOW and SEARCH_HIGH or to SEARCH_HIGH, reduced by a reasonable increments for tunnel headers.

If SEARCH_LOW corresponds to a common link MTU, set the candidate MPS to SEARCH_LOW plus some small delta. If this fails, we found the proper MPS, otherwise we need to keep searching.

@@@@@ common link MTUs are: 1500..... ?

@@@@@ common tunnel header sizes are....

7.6 Congestion Control and Window Management

PLPMTUD and congestion control share the same slice of the protocol stack. Both algorithms nominally run inside of a transport protocol and rely on packet losses as their primary signal to adjust parameters of the data stream (packet size or window size). Furthermore both push up the controlled parameter until the onset of packet losses, and then back off to a smaller value. Due to the close proximity of these two algorithms there is the potential for side effects and unexpected interactions between them.

This section describes potential interactions between PLPMTUD and congestion control. In general PLPMTUD is designed to minimize its potential impact on congestion control. This is appropriate because correctly functioning congestion control is critical to the overall operation of the Internet.

The requirements in section 4 protect congestion control from PLPMTUD. It is important that MTU changes do not raise the congestion window. Given that we do not know a priori the nature of the network bottleneck, PLPMTUD should not raise either the data rate (bytes per second) or the packet rate (packets per second).

Since there is a risk that lost probes might actually be congestion losses, and not MTU losses at all, we limit the maximum allowed rate for suppressing congestion control to less than the loss rate required to throttle the flow to the "TCP friendly" rate. This guarantees that the losses due to PLPMTUD are less than the losses needed for normal congestion control.

If there is some node which is accounting queue length in bytes

(rather than packets), there is even the possibility that a probe might cause a loss due to driving the queue over some threshold and into congestion. For this reason it recommended that all PLPMTUD implementations use some strategy to slightly depress the actual window during the probe process. It may be sufficient to require that the excess data in the probe packet fits within the current congestion control window.

If a probe is carrying real application data that must be retransmitted, it is important to suppress (or restore) all of the congestion control state changes normally associated with the retransmission. For example if a TCP connection is in slow-start when a probe is lost, it is important that ssthresh is not changed as a side effect of the probing. It is for this reason that it is strongly recommended that packetization protocols use some combination of out-of-band echo message and padding, if at all possible. Lost probes that do not carry any real application data do not need to be retransmitted.

It is recommended that TCP should not probe a new MPS if that MPS will likely result in a cwnd of less than 5 segments.

If the network becomes too congested, it is recommended that the MPS be reduced to a smaller size as determined by a heuristic. The recommended heuristic is to reduce the MPS by half if ssthresh is reduced to 5 segments or smaller, with a minimum MPS of 512 bytes.

8. 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 7. 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.

8.1 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 the overheads 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 need 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 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 implementation it can be relatively expensive to adjust segment boundaries after the data is already queued.

If TCP is using SACK or any other variable length headers, the headers on the probe and verification packets should be padded to the maximum possible length. Otherwise, unexpected options on bidirectional data may cause cause IP packets that are larger than the tested MTU.

At the point when TCP is ready to start the verification phase, it is permitted transmit already queued data at the old MTU rather than re-packetizes it. This postpones the verification process by the time required to send the queued data.

If the verification phase experiences any segment losses, TCP is required to pull back to the prior MSS. Since failing the verification phase should be an infrequent error condition it is less important that this be as efficient as probing.

8.2 Probing method using SCTP

In the SCTP protocol [9][16] 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 TSN's. 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 MPS (maximum payload size, same as MTU - IP headers) of all paths. Although not required, implementations may bundle multiple data chunks together to make larger IP packets to allow for support for larger MPSs on different paths. Note that SCTP must independently probe and verify the MPS 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 the returning the same Heartbeat as a HEARTBEAT-ACK. This method is fully compatible with current SCTP standards and implementations, but is exposed to MPS 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 discard 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 types 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 MPS are not applied to the forward path.

The verification phase is most efficiently implemented by picking a new chunk size such that the new MPS and all of the old multi-path MPSs are larger than different multiples of the new chunk size, by at least the required header sizes. This approach permits chunks from SCTP application messages to be assembled into packets that are suitable for any path to the peer at either the old or new MPS. This is the easiest method to permit the provisional MPS to be withdrawn, if there are losses during the verification phase.

Once each of old path MPSs has been updated to a new verified MPS, SCTP may be able to pick a new larger chunk size that will fit into all paths. However, if the MPS is later reduced (say due to a routing change and subsequent ICMP PTB message) SCTP will be forced to use IP fragmentation to transmit application messages that are already chunkified, as described in section 7.3.

The constraints on efficiently choosing chunk sizes are complicated enough to make it difficult if not impossible to efficiently support arbitrary combinations of old and new MPSs. It greatly simplify the implementation to add constraints, such as making the chunk size itself a multiple of some common size, such as 512 bytes. This in turn constrains the searching algorithm to test MPSs that are multiples of 512 bytes, plus the appropriate headers. Clearly the

PLPMTUD search heuristic for SCTP must be constrained to pick candidate MPSs that are consistent with the limitations of the algorithm for choosing appropriate chunk sizes.

The SCTP Verification-Tag is designed to increase SCTPs robustness in the presence of a number of attacks, including forged ICMP messages. It relies on a 32 bit Verification Tag which is initialized to a random value during connection establishment and placed in the first 64 bits of all SCTP messages. All subsequent messages (including ICMP messages, which copy at least the first 64 bits of the message) must match the original Verification Tag, or they are rejected as being likely attacks against the connection.

It is believed that the Verification Tag mechanism is strong enough where SCTP could unconditionally process ICMP PTB messages that would reduce the path MPS at arbitrary times. As written, this document does not encourage this method. The PLPMTUD ICMP validity checks are cascaded with the SCTP checks, such that the messages are processed only if they meet all consistency checks for both protocols. In particular, PLPMTUD only uses the ICMP MPS value following a probe, during MPS verification, or following a full stop timeout.

Alternatively, an SCTP implementation could suppress some of the checks in section 6.2.1.

8.3 Probing method for IP fragmentation

As mentioned in section 6.4, 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 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 MPS should be selected as described in section 7.4, except with a separate global control for the default initial MPS 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 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. We take this approach: 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 MPS that is fixed at the initial value. A new path cache entry is not created until there is an attempt to set the MPS.

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 9.4 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 losses can prevent PLPMTUD from raising the MTU, forcing the Packetization Layer protocol to use a smaller MTU than necessary. Since these failures are not likely to cause interoperability problem 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 likely situation.

8.4 Probing method for applications

The disadvantages of probing with ICMP ECHO can be overcome by implementing the path MTU discovery daemon within the application itself, using applications 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) has is preferred because you can send large probes that causes small acknowledgments. For protocols that can not implement these messages directly there are often alternate methods for generating probes. E.g 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 to others 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 issues: 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 can send probes that are larger than the IP layers 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.

9. Operational Integration

This section describes ways to minimize deployment problems for

PLPMTUD, by including a number of good management features: mechanisms to diagnose problems with path MTU discovery, and configuration controls such that the more risky properties can be progressively deployed. We also address some potentially serious interactions with nodes that do not honor the DF bit.

9.1 Interoperation with prior algorithms

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. Therefore, we offer a deployment strategy in which classical PMTUD operation as described in RFC 1191 and RFC 1981 is unmodified and PLPMTUD is only invoked following a full stop timeout, presumably due to an "ICMP black hole". To do this:

- Relax the ICMP checks in section 6.2.1 specifically to allow an ICMP Packet Too Large message to reduce the MTU at arbitrary times.
- When there is no cached MTU, use the Interface MTU as specified by classical PMTU discovery, rather the initial MTU as specified in section 7.4
- MTU searching as described in section 7.5 is disabled by setting SEARCH_HIGH equal to SEARCH_LOW and the initial MPS.
- A full stop timeout is processed as described in section 6.2.4. This becomes the only mechanism to invoke the rest of PLPMTUD.

When configured in this manner, PLPMTUD will increase the robustness of classical PMTU discovery in the presence of ICMP black holes and other ICMP problems, with minimal exposure to unanticipated problems during deployment. Since this configuration does not help robustness in the presence of malicious or erroneous ICMP messages, it is not recommended for the long term.

9.2 Operation over subnets with dissimilar MTUs

With classical PMTUD, the ingress router to a subnet is responsible for knowing what size packets can be delivered to every node attached to that subnets. For most subnet types, this requires that the entire subnet has a single MTU which is common to every attached node. (For a few subnets types, such as ATM[12] the nodes on a subnet can negotiate the MTU on a pairwise basis, and the ingress router is responsible for knowing the MTU to each of it peers).

This requirement has proven to be a major impediment to deploying larger MTUs in the operational Internet. Often one single node which does not support a larger MTU effectively vetoes raising the MTU on a subnet, because the ingress router does not have a mechanism to

generate the proper ICMP PTB message for the one attached node with a smaller MTU.

With PLPMTUD, this requirement is completely relaxed. As long as oversized packets addressed to nodes with the smaller MTU are reliably discarded, PLPMTUD will find the proper MTU for these nodes.

Once there sufficient field experience to demonstrate that PLPMTUD is robust, we recommend that OS vendors consider updating default MTUs for Network Interface Cards. It would raise the overall performance of the Internet if all NICs were configured to default to the MTU which is most efficient for the NIC (lowest overhead per byte), rather than the standard MTU for the media or switch. This is most likely to be the largest MTU supported by the NIC chip set or some other logical boundary, such as memory page sizes.

9.3 Interoperation with tunnels

PLPMTUD is specifically designed to solve many of the problems that people are experiencing today due to poor interactions between classical MTU discovery, IPsec, and various sorts of tunnels [5]. As long as the tunnel reliably discards packets that are too large, PLPMTUD will discover an appropriate MTU for the path.

Unfortunately due to the pervasive problems with classical PMTU discovery, many manufacturers of various types of VPN/tunneling equipment have resorted to ignoring the DF bit under some conditions. This not only violates the IP standard and many recommendations to the contrary [17][18], it also violates the only requirement that PLPMTUD places on the link layer: that oversized packets are reliably discarded. It is imperative that people understand the impact of ignoring the DF bit both to applications and to PLPMTUD.

We do understand the reality of the situation. It is important that vendors who are building devices the violate the DF specification understand that PLPMTUD requires that probe packets be discarded, and that sending ICMP PTB messages alone is insufficient to prevent wholesale fragmentation if the probe packets are delivered.

Therefore, it is imperative that devices that do not honor DF include packet size history caches and other heuristics to robustly detect and discard probe packets, if delivering them would require fragmentation.

9.4 Diagnostic tools

All implementations MUST include facilities for MTU discovery diagnostic tools that implement PLPMTUD or other MTU discovery

algorithms in user mode without help or interference by the PMTUD algorithm present in the operating system. This requires an 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 the MTU discovery, such as routers that return ICMP PTB messages containing incorrect size information.

9.5 Management interface

It is suggested that an implementation provide a way for a system utility program to:

- Globally disable all ICMP Packet Tool Large message processing
- Globally suppress some or all ICMP consistency checks described in section 6.2.1. Setting this option fore goes some possible security improvements, in exchange for making PLPMTUD behave more like classical PMTU discovery. (See section 9.1)
- Globally permit ICMP Packet Tool Large messages to unconditionally reduce the MTU, even if there were not lost lost packets. Setting option fore goes some possible security improvements, in exchange for making PLPMTUD behave more like classical PMTU discovery. (See section 9.1)
- Globally adjust timer intervals for specific classes of probe failures

In addition, it is important that there be a mechanism to permit per path controls to override specific parts of the PLPMTUD algorithm. All of these per path controls should be preset from similar global controls:

- Disable MTU searching a given path, such that new MTU values are never probed.
- Set the initial MTU for a given path. This could be used to speed convergence in relatively static environments. There should be an option to cause PLPMTUD to choose the same initial value as would be chosen by classical PMTU discovery. I.e. typically the Interface MTU. This is used in the mode described in section 9.1 where PLPMTUD is used only for black hole detection in classical PMTU discovery.
- Limit the maximum probed MTU for a given path. This permits a manual configuration to work around a link that spuriously delivers packets that are larger than the useful path MTU.

- Per path and per application controls to disable ICMP processing, to further limit possible damage from malicious ICMP PTB messages (in addition to the global controls).

10. References

10.1 Normative References

[1] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[2] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, November 1990.

[3] McCann, J., Deering, S. and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, August 1996.

[4] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[5] Kent, S. and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[6] Allman, M., Floyd, S. and C. Partridge, "Increasing TCP's Initial Window", RFC 2414, September 1998.

[7] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[8] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, September 2000.

[9] Stewart, R., Xie, Q., Morneault, K., Sharp, C., Schwarzbauer, H., Taylor, T., Rytina, I., Kalla, M., Zhang, L. and V. Paxson, "Stream Control Transmission Protocol", RFC 2960, October 2000.

10.2 Informative References

[10] Mogul, J., Kent, C., Partridge, C. and K. McCloghrie, "IP MTU discovery options", RFC 1063, July 1988.

[11] Knowles, S., "IESG Advice from Experience with Path MTU Discovery", RFC 1435, March 1993.

[12] Atkinson, R., "Default IP MTU for use over ATM AAL5", RFC 1626, May 1994.

[13] Sung, T., "TCP And UDP Over IPX Networks With Fixed Path MTU", RFC 1791, April 1995.

[14] Partridge, C., "Using the Flow Label Field in IPv6", RFC 1809, June 1995.

[15] Lahey, K., "TCP Problems with Path MTU Discovery", RFC 2923, September 2000.

[16] Stewart, R., "Stream Control Transmission Protocol (SCTP) Implementors Guide", draft-ietf-tsvwg-sctpimpguide-10 (work in progress), December 2003.

[17] Kent, C. and J. Mogul, "Fragmentation considered harmful", Proc. SIGCOMM '87 vol. 17, No. 5, October 1987.

[18] Mathis, M., Heffner, J. and B. Chandler, "Fragmentation Considered Very Harmful", draft-mathis-frag-harmful-00 (work in progress), July 2004.

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

Kevin Lahey
Freelance

EMail: kml@patheticgeek.net

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].

It the recommended configuration, PLPMTUD is significantly harder to attack than current procedures, because ICMP messages are cached and only processed in connection with lost packets. This effectively prevents blind attacks on the path MTU discovery system.

Furthermore, since this algorithm is designed for robust operation without any ICMP (or other messages from the network), it can be configured to ignore all ICMP messages (globally or on a per application basis). In this configuration it can not 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.

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

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