Network Working Group
Request for Comments: 4821
Category: Standards Track

M. Mathis
J. Heffner
PSC

March 2007

Packetization Layer Path MTU Discovery

Status of This Memo

Copyright Notice

Abstract

This document describes a robust method for Path MTU Discovery (PMTUD) 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.

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 described in [RFC1191] and [RFC1981]. In the absence of ICMP messages, 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 does not update RFC 1191 or RFC 1981; however, since it supports correct operation without ICMP, it implicitly relaxes some of the requirements for the algorithms specified in those documents.

The methods described in this document rely on features of existing protocols. They apply to many transport protocols over IPv4 and IPv6. They do not require cooperation from the lower layers (except that they are consistent about which packet sizes are acceptable) or from peers. As the methods apply only to senders, variants in implementations will not cause interoperability problems.

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 [RFC2119].

This document is a product of the Path MTU Discovery (PMTUD) working group of the IETF and draws heavily on RFC 1191 and RFC 1981 for terminology, ideas, and some of the text.

2. Overview

Packetization Layer Path MTU Discovery (PLPMTUD) is a method for TCP or other Packetization Protocols to dynamically discover the MTU of a path by probing with progressively larger packets. It is most efficient when used in conjunction with the ICMP-based Path MTU Discovery mechanism as specified in RFC 1191 and RFC 1981, but resolves many of the robustness problems of the classical techniques since it does not depend on the delivery of ICMP messages.

This method is 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 Too 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 [RFC2914]. Probing can resume after a delay that 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 lower limit on a successful probe or lowering the upper limit on a failed probe, converging toward the true Path MTU. For most transport layers, the search should be stopped once the range is narrow enough that the benefit of a larger effective Path MTU is smaller than the search overhead of finding it.

The most likely (and least serious) probe failure is due to 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 process can be used to detect 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.

Packetization Layer PMTUD (PLPMTUD) introduces some flexibility in the implementation of classical Path MTU Discovery. It can be configured to perform just ICMP black hole recovery to increase the robustness of classical Path MTU Discovery, or at the other extreme, all ICMP processing can be disabled and PLPMTUD can completely replace classical Path MTU Discovery.

Classical Path MTU Discovery is subject to protocol failures (connection hangs) if ICMP Packet Too Big (PTB) messages are not delivered or processed for some reason [RFC2923]. With PLPMTUD, classical Path MTU Discovery can be modified to include additional consistency checks without increasing the risk of connection hangs due to spurious failures of the additional 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 to discover the Path MTU. In this configuration, PLPMTUD parallels congestion control. An end-to-end transport protocol adjusts 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 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.

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.

This document does not contain a complete description of an implementation. It only sketches details that do not affect interoperability with other implementations and have strong externally imposed optimality criteria (e.g., the MTU searching and

caching heuristics). Other details are explicitly included because there is an obvious alternative implementation that doesn't work well in some (possibly subtle) case.

Section 3 provides a complete glossary of terms.

Section 4 describes the details of PLPMTUD that affect interoperability with other standards or Internet protocols.

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 describes how to use probes to search for the Path MTU.

Section 8 recommends using IPv4 fragmentation in a configuration that mimics IPv6 functionality, to minimize future problems migrating to IPv6.

Section 9 describes a programming interface for implementing PLPMTUD in applications that choose their own packet boundaries 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:

IP: Either IPv4 [RFC0791] or IPv6 [RFC2460].

Node: A device that implements IP.

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 Asynchronous Transfer Mode (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. Be aware that this definition is different 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 that 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 that 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 "Fragmentation Needed and DF Set" message.

Flow: A context in which MTU Discovery algorithms can be invoked. This is naturally an instance of a Packetization Protocol, for example, one side of a TCP connection.

MSS: The TCP Maximum Segment Size [RFC0793], 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 that 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, including IP headers.

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 7.3.

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. This is described in more detail in Section 7.7.

4. Requirements

All links MUST enforce their MTU: links that might non-deterministically deliver packets that are larger than their rated MTU MUST consistently discard such packets.

In the distant past, there were a small number of network devices that did not enforce MTU, but could not reliably deliver oversized packets. For example, some early bit-wise Ethernet repeaters would forward arbitrarily sized packets, but could not do so reliably due to finite hardware data clock stability. This is the only requirement that PLPMTUD places on lower layers. It is important that this requirement be explicit to forestall the future standardization or deployment of technologies that might be incompatible with PLPMTUD.

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 8.

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 SHOULD be suppressed. If any other data loss is detected, standard congestion control MUST take place.

Suppressed congestion control 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 [tcp-friendly]). 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. This is discussed further in Section 7.6.2.

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 so as 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 (as described in Section 5.2) SHOULD be notified to make use of the new MTU and make the required congestion control adjustments.

All implementations MUST include mechanisms for applications to selectively transmit packets larger than the current effective Path MTU, but smaller than the first-hop link MTU. This is necessary to implement PLPMTUD using a connectionless protocol within an application and to implement diagnostic tools that do not rely on the operating system's implementation of Path MTU Discovery. See Section 9 for further discussion.

Implementations MAY use different heuristics to select the initial effective Path MTU for each protocol. Connectionless protocols and protocols that do not support PLPMTUD SHOULD have their own default value for the initial effective Path MTU, which can be set to a more conservative (smaller) value than the initial value used by TCP and other protocols that are well suited to PLPMTUD. There SHOULD be per-protocol and per-route limits on the initial effective Path MTU (eff_pmtu) and the upper searching limit (search_high). See Section 7.2 for further discussion.

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 and verification of 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 a direct extension of the advice in the current PMTUD specifications in RFC 1191 and RFC 1981.

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 conducting a probe, the Packetization Layer MUST determine the probe packet's final size including IP headers. This requirement is specific to PLPMTUD, and satisfying it may require additional inter-layer communication in existing implementations.

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 (eff_pmtu in Section 7.1) MAY be shared between different flows with a common path representation.

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

Within a single implementation, different protocols can use either of these two approaches. Due to protocol specific differences in constraints on generating probes (Section 6.2) and the MTU searching algorithm (Section 7.3), 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 other protocols can only share observed PMTU. In this case, the different protocols will have different PMTU convergence properties.

The IP layer SHOULD be used to store the cached PMTU value 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 MAY 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, and integrates nicely with the conceptual model of a host as described in [RFC2461]: a PMTU value could be stored with the corresponding entry in the destination cache. Since Network Address Translators (NATs) and other forms of middle boxes may exhibit differing PMTUs simultaneously at a single IP address, the minimum value SHOULD be stored.

Network or subnet numbers MUST NOT be used as representations of a path, because there is not a general mechanism to determine the network mask at the remote host.

For source-routed packets (i.e., packets containing an IPv6 routing header, or IPv4 Loose Source and Record Route (LSRR) or Strict Source and Record Route (SSRR) options), the source route MAY further qualify the local representation of a path. An implementation MAY use source route information in the local representation of a path.

If IPv6 flows are in use, an implementation MAY use the 3-tuple of the Flow label and the source and destination addresses [RFC2460][RFC3697] as the local representation of a path. Such an approach could theoretically 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.

5.3. Accounting for IPsec

This document does not take a stance on the placement of IP Security (IPsec) [RFC2401], which logically sits between IP and the Packetization Layer. A 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. 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.

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 MAY maintain a single MTU value to be used for all multicast packets originated from the node. This MTU SHOULD be sufficiently small that it is expected to be less than the Path MTU of all paths comprising the multicast tree. If a Path MTU of less than the configured multicast MTU is learned via unicast means, the multicast MTU MAY be reduced to this value. 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 since this requirement has poor scaling properties), PLPMTUD MAY be implemented in multicast protocols such that the smallest path MTU learned across a group becomes the effective MTU for that group.

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 important that the mechanism can robustly distinguish between the isolated loss of just a probe and other losses in the probe's leading and trailing windows.

It is best if Packetization Protocols use an explicit loss detection mechanism such as a Selective Acknowledgment (SACK) scoreboard [RFC3517] or ACK Vector [RFC4340] to distinguish real losses from reordered data, although implicit mechanisms such as TCP Reno style duplicate acknowledgments counting are sufficient.

PLPMTUD can also be implemented in protocols that rely on timeouts as their primary mechanism for loss recovery; however, timeouts SHOULD NOT be used as the primary mechanism for loss indication unless 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 ensured 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 PAD chunk 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 be no reasonable mechanisms to generate probes within the Packetization Layer protocol itself. As a last resort, it may be possible to rely on an adjunct protocol, such as ICMP ECHO ("ping"), to send probe packets. See Section 10.3 for further discussion of this approach.

7. 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.

7.1. Packet Size Ranges

This document describes the probing method using three state variables:

search_low: The smallest useful probe size, minus one. The network is expected to be able to deliver packets of size search_low.

search_high: The greatest useful probe size. Packets of size search_high are expected to be too large for the network to deliver.

eff_pmtu: The effective PMTU for this flow. This is the largest non-probe packet permitted by PLPMTUD for the path.

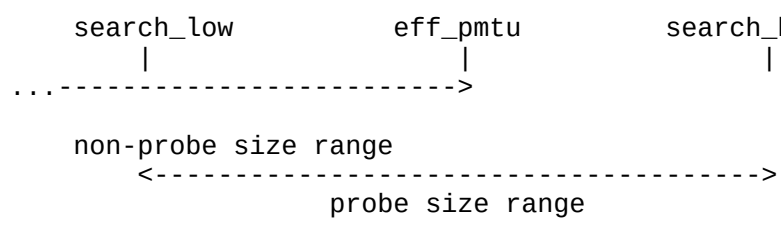

Figure 1

When transmitting non-probes, the Packetization Layer SHOULD create packets of a size less than or equal to eff_pmtu.

When transmitting probes, the Packetization Layer MUST select a probe size that is larger than search_low and smaller than or equal to search_high.

When probing upward, eff_pmtu always equals search_low. In other states, such as initial conditions, after ICMP PTB message processing or following PLPMTUD on another flow sharing the same path representation, eff_pmtu may be different from search_low. Normally, eff_pmtu will be greater than or equal to search_low and less than search_high. It is generally expected but not required that probe size will be greater than eff_pmtu.

For initial conditions when there is no information about the path, eff_pmtu may be greater than search_low. The initial value of search_low SHOULD be conservatively low, but performance may be better if eff_pmtu starts at a higher, less conservative, value. See Section 7.2.

If eff_pmtu is larger than search_low, it is explicitly permitted to send non-probe packets larger than search_low. When such a packet is acknowledged, it is effectively an "implicit probe" and search_low SHOULD be raised to the size of the acknowledged packet. However, if an "implicit probe" is lost, it MUST NOT be treated as a probe failure as a true probe would be. If eff_pmtu is too large, this condition will only be detected with ICMP PTB messages or black hole discovery (see Section 7.7).

7.2. Selecting Initial Values

The initial value for search_high SHOULD be the largest possible packet that might be supported by the flow. This may be limited by the local interface MTU, by an explicit protocol mechanism such as the TCP MSS option, or by an intrinsic limit such as the size of a protocol length field. In addition, the initial value for search_high MAY be limited by a configuration option to prevent probing above some maximum size. Search_high is likely to be the same as the initial Path MTU as computed by the classical Path MTU Discovery algorithm.

It is RECOMMENDED that search_low be initially set to an MTU size that is likely to work over a very wide range of environments. Given today's technologies, a value of 1024 bytes is probably safe enough. 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 causes any unexpected changes in protocol behaviors. The selection of the initial value for eff_pmtu determines to what extent a PLPMTUD implementation's behavior resembles classical PMTUD in cases where the classical method is sufficient.

A conservative configuration would be to set eff_pmtu to search_high, and rely on ICMP PTB messages to set the eff_pmtu down as appropriate. In this configuration, classical PMTUD is fully functional and PLPMTUD is only invoked to recover from ICMP black holes through the procedure described in Section 7.7.

In some cases, where it is known that classical PMTUD is likely to fail (for example, if ICMP PTB messages are administratively disabled for security reasons), using a small initial eff_pmtu will avoid the costly timeouts required for black hole detection. The trade-off is that using a smaller than necessary initial eff_pmtu might cause reduced performance.

Note that the initial eff_pmtu can be any value in the range search_low to search_high. An initial eff_pmtu of 1400 bytes might be a good compromise because it would be safe for nearly all tunnels over all common networking gear, and yet close to the optimal MTU for the majority of paths in the Internet today. This might be improved by using some statistics of other recent flows: for example, the initial eff_pmtu for a flow might be set to the median of the probe size for all recent successful probes.

Since the cost of PLPMTUD is dominated by the protocol specific overheads of generating and processing probes, it is probably desirable for each protocol to have its own heuristics to select the initial eff_pmtu. It is especially important that connectionless protocols and other protocols that may not receive clear indications of ICMP black holes use conservative (smaller) initial values for eff_pmtu, as described in Section 10.3.

There SHOULD be per-protocol and per-route configuration options to override initial values for eff_pmtu and other PLPMTUD state variables.

7.3. 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, such as TCP, failed probes are more expensive than successful ones, since data in a failed probe will need to be retransmitted. For such protocols, a strategy that raises the probe size in smaller increments might have lower overhead. 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 use other mechanisms to choose the probe sizes. For example, protocols that have certain natural data block sizes might simply assemble messages from a number of blocks until the total size is smaller than search_high, and if possible larger than search_low.

Each Packetization Layer MUST determine when probing has converged, that is, when the probe size range is small enough that further probing is no longer worth its cost. When probing has converged, a timer SHOULD be set. When the timer expires, search_high should be reset to its initial value (described above) so that probing can resume. Thus, if the path changes, increasing the Path MTU, then the flow will eventually take advantage of it. The value for this timer MUST NOT be less than 5 minutes and is recommended to be 10 minutes, per RFC 1981.

7.4. Probing Preconditions

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

- It has no outstanding probes or losses.
- If the last probe failed or was inconclusive, then the probe timeout has expired (see Section 7.6.2).
- The available window is greater than the probe size.
- For a protocol using in-band data for probing, enough data is available to send the probe.

In addition, the timely loss detection algorithms in most protocols have pre-conditions that SHOULD be satisfied before sending a probe. For example, TCP Fast Retransmit is not robust unless there are sufficient segments following a probe; that is, the sender SHOULD have enough data queued and sufficient receiver window to send the probe plus at least Tcprexmtthresh [RFC2760] additional segments. This restriction may inhibit probing in some protocol states, such as too close to the end of a connection, or when the window is too small.

Protocols MAY delay sending non-probes in order to accumulate enough data to meet the pre-conditions for probing. The delayed sending algorithm SHOULD use some self-scaling technique to appropriately limit the time that the data is delayed. For example, the returning ACKs can be used to prevent the window from falling by more than the amount of data needed for the probe.

7.5. 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 to the probe size. After sending the probe it awaits a response, which will have one of 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 whether 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.

7.6. Response to Probe Results

When a probe has completed, the result SHOULD be processed as follows, categorized by the probe's result type.

7.6.1. Probe Success

When the probe is delivered, it is an indication that the Path MTU is at least as large as the probe size. Set search_low to the probe size. If the probe size is larger than the eff_pmtu, raise eff_pmtu to the probe size. The probe size might be smaller than the eff_pmtu if the flow has not been using the full MTU of the path because it is subject to some other limitation, such as available data in an interactive session.

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 7.8.

7.6.2. Probe Failure

When only the probe is lost, it 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, set search_high to the probe size minus one. The eff_pmtu might be larger than the probe size if the flow has not been using the full MTU of the path because it is subject to some other limitation, such as available data in an interactive session. If eff_pmtu is larger than the probe size, eff_pmtu MUST be reduced to no larger than search_high, and SHOULD be reduced to search_low, as the eff_pmtu has been determined to be invalid, similar to after a full-stop timeout (see Section 7.7).

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 SHOULD ignore 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 MUST NOT probe again until an interval that is larger than the expected interval between congestion control events. See Section 4 for details. 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.

7.6.3. 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 time interval until the next probe. A time interval that is five times the non-timeout failure case (Section 7.6.2) is RECOMMENDED.

7.6.4. 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 due to an MTU limitation. In this case, the state variables eff_pmtu, search_low, and search_high SHOULD NOT be updated, and the same-sized probe SHOULD be attempted again as soon as the probing preconditions are met (i.e., once the packetization layer has no outstanding unrecovered losses). 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.

7.7. 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 [RFC1122] 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 message was ignored for some reason), the RECOMMENDED first recovery action is to treat this as a detected ICMP black hole as defined in [RFC2923].

The response to a detected black hole depends on the current values for search_low and eff_pmtu. If eff_pmtu is larger than search_low, set eff_pmtu to search_low. Otherwise, set both eff_pmtu and search_low to the initial value for search_low. Upon additional successive timeouts, search_low and eff_pmtu SHOULD be halved, with a lower bound of 68 bytes for IPv4 and 1280 bytes for IPv6. Even lower lower bounds MAY be permitted to support limited operation over links with MTUs that are smaller than permitted by the IP specifications.

7.8. MTU Verification

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

If ICMP delivery fails, breaking classical PMTUD, the connection will rely solely on PLPMTUD. In this case, PLPMTUD may fail as well since it assumes a flow traverses a path with a single MTU. 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 will significantly increase. The flow may still make progress, but the resultant loss rate is likely to be unacceptable. For example, when using two-way round-robin striping, 50% of full-sized packets would be dropped.

Striping in this manner is often operationally undesirable for other reasons (e.g., due to packet reordering) and is usually avoided by hashing each flow 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 to save the value of eff_pmtu before raising it. Then, if loss rate rises above a threshold for a period of time (e.g., loss rate is higher than 10% over multiple retransmission timeout (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. PLPMTUD implementations SHOULD implement MTU verification.

8. Host Fragmentation

Packetization Layers SHOULD avoid sending messages that will require fragmentation [Kent87] [frag-errors]. 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 an 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 effective Path MTU, they SHOULD be fragmented to the Path MTU in the host IP layer even if they are smaller than the MTU of the first link, 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 the likelihood that applications will rely on IPv4 fragmentation in a way that cannot be implemented in IPv6. At least one major operating system already uses this strategy. Section 9 describes some exceptions to this rule when the application is sending oversized packets for probing or diagnostic purposes.

Since protocols that do not implement PLPMTUD are still subject to problems due to ICMP black holes, it may be desirable to limit to these protocols to "safe" MTUs likely to work on any path (e.g., 1280 bytes). Allow any protocol implementing PLPMTUD to operate over the full range supported by the lower layer.

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 cannot support a native implementation of PLPMTUD. Fragmentation-based PLPMTUD requires an adjunct protocol as described in Section 10.3.

9. Application Probing

All implementations MUST include a mechanism where applications using connectionless protocols can send their own probes. This is necessary to implement PLPMTUD in an application protocol as described in Section 10.4 or to implement diagnostic tools for debugging problems with PMTUD. There MUST be a mechanism that permits an application to send datagrams that are larger than eff_pmtu, the operating systems estimate of the Path MTU, without being fragmented. If these are IPv4 packets, they MUST have the DF bit set.

At this time, most operating systems support two modes for sending datagrams: one that silently fragments packets that are too large, and another that rejects packets that are too large. Neither of these modes is suitable for implementing PLPMTUD in an application or diagnosing problems with Path MTU Discovery. A third mode is REQUIRED where the datagram is sent even if it is larger than the current estimate of the Path MTU.

Implementing PLPMTUD in an application also requires a mechanism where the application can inform the operating system about the outcome of the probe as described in Section 7.6, or directly update search_low, search_high, and eff_pmtu, described in Section 7.1.

Diagnostic applications are useful for finding PMTUD problems, such as those that might be caused by a defective router that returns ICMP PTB messages with incorrect size information. Such problems can be most quickly located with a tool that can send probes of any specified size, and collect and display all returned ICMP PTB messages.

10. Specific Packetization Layers

All Packetization Layer protocols must consider all of the issues discussed in Section 6. For many protocols, it is straightforward to address these issues. This section discusses specific details for implementing PLPMTUD with a couple of protocols. It is hoped that the descriptions here will be sufficient illustration for implementers to adapt to additional protocols.

10.1. Probing Method Using TCP

TCP has no mechanism to distinguish in-band data from padding. Therefore, TCP must generate probes by appropriately segmenting data. There are two approaches to segmentation: overlapping and non-overlapping.

In the non-overlapping method, data is segmented such that the probe and any subsequent segments contain no overlapping data. If the probe is lost, the "probe gap" will be a full probe size minus headers. Data in the probe gap will need to be retransmitted with multiple smaller segments.

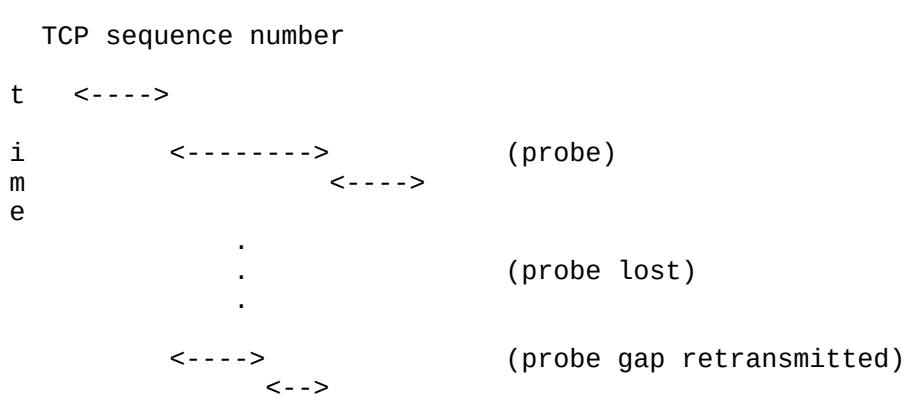

Figure 2

An alternate approach is to send subsequent data overlapping the probe such that the probe gap is equal in length to the current MSS. In the case of a successful probe, this has added overhead in that it will send some data twice, but it will have to retransmit only one segment after a lost probe. When a probe succeeds, there will likely be some duplicate acknowledgments generated due to the duplicate data sent. It is important that these duplicate acknowledgments not trigger Fast Retransmit. As such, an implementation using this approach SHOULD limit the probe size to three times the current MSS (causing at most 2 duplicate acknowledgments), or appropriately adjust its duplicate acknowledgment threshold for data immediately after a successful probe.

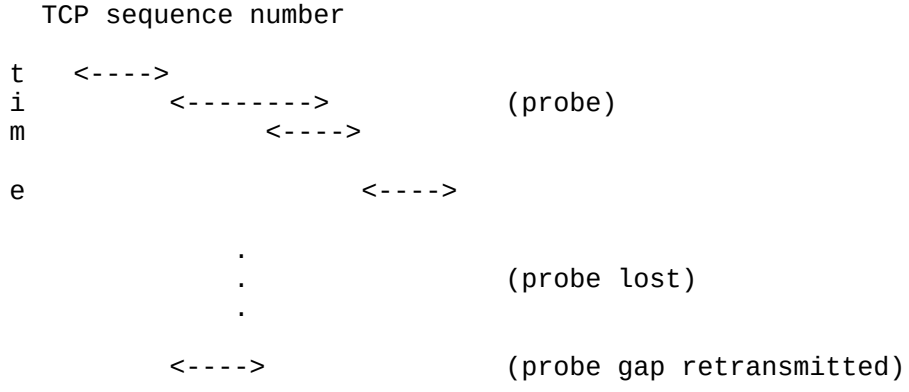

Figure 3

The choice of which segmentation method to use should be based on what is simplest and most efficient for a given TCP implementation.

10.2. Probing Method Using SCTP

In the Stream Control Transmission Protocol (SCTP) [RFC2960], the application writes messages to SCTP, which divides the data into smaller "chunks" suitable for transmission through the network. Each chunk is assigned a Transmission Sequence Number (TSN). Once a TSN has been transmitted, SCTP cannot change the chunk size. SCTP multi-path support normally requires SCTP to choose a chunk size such that 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. The PAD chunk defined in [RFC4820] SHOULD be attached to a minimum length HEARTBEAT (HB) chunk to build a probe packet. This method is fully compatible with all current SCTP implementations.

SCTP MAY also probe with a method similar to TCP's described above, using inline data. Using such a method has the advantage that successful probes have no additional overhead; however, failed probes will require retransmission of data, which may impact flow performance.

10.3. Probing Method for IP Fragmentation

There are a few protocols and applications that normally send large datagrams and rely on IP fragmentation to deliver them. It has been known for a long time that this has some undesirable consequences [Kent87]. More 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 mis-associated IP fragments, and the TCP and UDP checksums are insufficient to prevent the resulting corrupted data from being delivered to higher protocol layers [frag-errors].

As mentioned in Section 8, datagram protocols (such as UDP) might rely on IP fragmentation as a Packetization Layer. However, using IP fragmentation to implement PLPMTUD is problematic because the IP layer has no mechanism to determine whether the packets are ultimately delivered to the far node, without direct participation by the application.

To support IP fragmentation as a Packetization Layer under an unmodified application, an implementation SHOULD rely on the Path MTU sharing described in Section 5.2 plus an adjunct protocol to probe the Path MTU. There are a number of protocols that might be used for the purpose, such as ICMP ECHO and ECHO REPLY, or "traceroute" style UDP datagrams that trigger ICMP messages. Use of ICMP ECHO and ECHO REPLY will probe both forward and return paths, so the sender will only be able to take advantage of the minimum of the two. Other methods that probe only the forward path are preferred if available.

All of these approaches have a number of potential robustness problems. The most likely failures are due to losses unrelated to MTU (e.g., nodes that discard some protocol types). These non-MTU-related losses can prevent PLPMTUD from raising the MTU, forcing IP fragmentation to use a smaller MTU than necessary. Since these failures are not likely to cause interoperability problems they are relatively benign.

However, other more serious failure modes do exist, such as might be caused by middle boxes or upper-layer routers that choose different paths for different protocol types or sessions. In such environments, adjunct protocols may legitimately experience a different Path MTU than the primary protocol. If the adjunct protocol finds a larger MTU than the primary protocol, PLPMTUD may select an MTU that is not usable by the primary protocol. Although this is a potentially serious problem, this sort of situation is likely to be viewed as incorrect by a large number of observers, and thus there will be strong motivation to correct it.

Since connectionless protocols might not keep enough state to effectively diagnose MTU black holes, it would be more robust to err on the side of using too small of an initial MTU (e.g., 1 kByte or less) prior to probing a path to measure the MTU. For this reason, implementations that use IP fragmentation SHOULD use an initial eff_pmtu, which is selected as described in Section 7.2, except using a separate global control for the default initial eff_mtu for connectionless protocols.

Connectionless protocols also introduce an additional problem with maintaining the path information cache: there are no events corresponding to connection establishment and tear-down to use to manage the cache itself. A natural approach would be to keep an immutable cache entry for the "default path", which has a eff_pmtu that is fixed at the initial value for connectionless protocols. The adjunct Path MTU Discovery protocol would be invoked once the number of fragmented datagrams to any particular destination reaches some configurable threshold (e.g., 5 datagrams). A new path cache entry would be created when the adjunct protocol updates eff_pmtu, and deleted on the basis of a timer or a Least Recently Used cache replacement algorithm.

10.4. Probing Method Using Applications

The disadvantages of relying on IP fragmentation and an adjunct protocol to perform Path MTU Discovery can be overcome by implementing Path MTU Discovery within the application itself, using the application's own protocol. The application must have some suitable method for generating probes and have an accurate and timely mechanism to determine whether the probes were lost.

Ideally, the application protocol includes a lightweight echo function that confirms message delivery, plus a mechanism for padding the messages out to the desired probe size, such that the padding is not echoed. This combination (akin to the SCTP HB plus PAD) is RECOMMENDED because an application can separately measure the MTU of each direction on a path with asymmetrical MTUs.

For protocols that cannot implement PLPMTUD with "echo plus pad", there are often alternate methods for generating probes. For example, the protocol may have a variable length echo that effectively measures minimum MTU of both the forward and return path's, or there may be a way to add padding to regular messages carrying real application data. There may also be alternate ways to segment application data to generate probes, or as a last resort, it may be feasible to extend the protocol with new message types specifically to support MTU discovery.

Note that if it is necessary to add new message types to support PLPMTUD, the most general approach is to add ECHO and PAD messages, which permit the greatest possible latitude in how an application-specific implementation of PLPMTUD interacts with other applications and protocols on the same end system.

All application probing techniques require the ability to send messages that are larger than the current eff_pmtu described in Section 9.

11. Security Considerations

Under all conditions, the PLPMTUD procedures described in this document are at least as secure as the current standard Path MTU Discovery procedures described in RFC 1191 and RFC 1981.

Since PLPMTUD is designed for robust operation without any ICMP or other messages from the network, it can be configured to ignore all ICMP messages, either globally or on a per-application basis. In such a configuration, it cannot be attacked unless the attacker can identify and cause probe packets to be lost. Attacking PLPMTUD reduces performance, but not as much as attacking congestion control by causing arbitrary packets to be lost. Such an attacker might do far more damage by completely disrupting specific protocols, such as DNS.

Since packetization protocols may share state with each other, if one packetization protocol (particularly an application) were hostile to other protocols on the same host, it could harm performance in the other protocols by reducing the effective MTU. If a packetization protocol is untrusted, it should not be allowed to write to shared state.

12. References

12.1. Normative References

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191, November 1990.

[RFC1981] McCann, J., Deering, S., and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, August 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC3697] Rajahalme, J., Conta, A., Carpenter, B., and S. Deering, "IPv6 Flow Label Specification", RFC 3697, March 2004.

[RFC2960] 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.

[RFC4820] Tuexen, M., Stewart, R., and P. Lei, "Padding Chunk and Parameter for the Stream Control Transmission Protocol (SCTP)", RFC 4820, March 2007.

12.2. Informative References

[RFC2760] Allman, M., Dawkins, S., Glover, D., Griner, J., Tran, D., Henderson, T., Heidemann, J., Touch, J., Kruse, H., Ostermann, S., Scott, K., and J. Semke, "Ongoing TCP Research Related to Satellites", RFC 2760, February 2000.

[RFC1122] Braden, R., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, October 1989.

[RFC2923] Lahey, K., "TCP Problems with Path MTU Discovery", RFC 2923, September 2000.

[RFC2401] Kent, S. and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, September 2000.

[RFC2461] Narten, T., Nordmark, E., and W. Simpson, "Neighbor Discovery for IP Version 6 (IPv6)", RFC 2461, December 1998.

[RFC3517] Blanton, E., Allman, M., Fall, K., and L. Wang, "A Conservative Selective Acknowledgment (SACK)-based Loss Recovery Algorithm for TCP", RFC 3517, April 2003.

[RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, March 2006.

[Kent87] Kent, C. and J. Mogul, "Fragmentation considered harmful", Proc. SIGCOMM '87 vol. 17, No. 5, October 1987.

[tcp-friendly] Mahdavi, J. and S. Floyd, "TCP-Friendly Unicast Rate-Based Flow Control", Technical note sent to the end2end-interest mailing list , January 1997, <http://www.psc.edu/networking/papers/tcp_friendly.html>.

[frag-errors] Heffner, J., "IPv4 Reassembly Errors at High Data Rates", Work in Progress, December 2007.

Appendix A. Acknowledgments

Many ideas and even some of the text come directly from RFC 1191 and RFC 1981.

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 versions. Thanks also to all of the people who made constructive comments in the working group meetings and on the mailing list. We are sure we 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
USA

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

Full Copyright Statement

Intellectual Property

Acknowledgement

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