Network Working Group
Request for Comments: 1981
Category: Standards Track

J. McCann
Digital Equipment Corporation
S. Deering
Xerox PARC
J. Mogul
Digital Equipment Corporation

August 1996

Path MTU Discovery for IP version 6

Status of this Memo

Abstract

This document describes Path MTU Discovery for IP version 6. It is largely derived from RFC 1191, which describes Path MTU Discovery for IP version 4.

Table of Contents

1. Introduction

When one IPv6 node has a large amount of data to send to another node, the data is transmitted in a series of IPv6 packets. It is usually preferable that these packets be of the largest size that can successfully traverse the path from the source node to the destination node. This packet size is referred to as the Path MTU (PMTU), and it is equal to the minimum link MTU of all the links in a path. IPv6 defines a standard mechanism for a node to discover the PMTU of an arbitrary path.

IPv6 nodes SHOULD implement Path MTU Discovery in order to discover and take advantage of paths with PMTU greater than the IPv6 minimum link MTU [IPv6-SPEC]. A minimal IPv6 implementation (e.g., in a boot ROM) may choose to omit implementation of Path MTU Discovery.

Nodes not implementing Path MTU Discovery use the IPv6 minimum link MTU defined in [IPv6-SPEC] as the maximum packet size. In most cases, this will result in the use of smaller packets than necessary, because most paths have a PMTU greater than the IPv6 minimum link MTU. A node sending packets much smaller than the Path MTU allows is wasting network resources and probably getting suboptimal throughput.

2. Terminology

node - a device that implements IPv6.

router - a node that forwards IPv6 packets not explicitly addressed to itself.

host - any node that is not a router.

upper layer - a protocol layer immediately above IPv6. 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) IPv6 such as IPX, AppleTalk, or IPv6 itself.

link - a communication facility or medium over which nodes can communicate at the link layer, i.e., the layer immediately below IPv6. 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 itself.

interface - a node’s attachment to a link.

address - an IPv6-layer identifier for an interface or a set of interfaces.

packet - an IPv6 header plus payload.

link MTU - the maximum transmission unit, i.e., maximum packet size in octets, that can be conveyed in one piece over a link.

path - the set of links traversed by a packet between a source node and a destination node

path MTU - the minimum link MTU of all the links in a path between a source node and a destination node.

PMTU - path MTU

Path MTU Discovery - process by which a node learns the PMTU of a path

flow - a sequence of packets sent from a particular source to a particular (unicast or multicast) destination for which the source desires special handling by the intervening routers.

flow id - a combination of a source address and a non-zero flow label.

3. Protocol overview

This memo describes a technique to dynamically discover the PMTU of a path. The basic idea is that a source node initially assumes that the PMTU of a path is the (known) MTU of the first hop in the path. If any of the packets sent on that path are too large to be forwarded by some node along the path, that node will discard them and return ICMPv6 Packet Too Big messages [ICMPv6]. Upon receipt of such a message, the source node reduces its assumed PMTU for the path based on the MTU of the constricting hop as reported in the Packet Too Big message.

The Path MTU Discovery process ends when the node's estimate of the PMTU is less than or equal to the actual PMTU. Note that several iterations of the packet-sent/Packet-Too-Big-message-received cycle may occur before the Path MTU Discovery process ends, as there may be links with smaller MTUs further along the path.

Alternatively, the node may elect to end the discovery process by ceasing to send packets larger than the IPv6 minimum link MTU.

The PMTU of a path may change over time, due to changes in the routing topology. Reductions of the PMTU are detected by Packet Too Big messages. To detect increases in a path's PMTU, a node periodically increases its assumed PMTU. This will almost always result in packets being discarded and Packet Too Big messages being generated, because in most cases the PMTU of the path will not have changed. Therefore, attempts to detect increases in a path's PMTU should be done infrequently.

Path MTU Discovery supports multicast as well as unicast destinations. In the case of a multicast destination, copies of a packet may traverse many different paths to many different nodes. Each path may have a different PMTU, and a single multicast packet may result in multiple Packet Too Big messages, each reporting a different next-hop MTU. The minimum PMTU value across the set of paths in use determines the size of subsequent packets sent to the multicast destination.

Note that Path MTU Discovery must be performed even in cases where a node "thinks" a destination is attached to the same link as itself. In a situation such as when a neighboring router acts as proxy [ND] for some destination, the destination can to appear to be directly connected but is in fact more than one hop away.

4. Protocol Requirements

As discussed in section 1, IPv6 nodes are not required to implement Path MTU Discovery. The requirements in this section apply only to those implementations that include Path MTU Discovery.

When a node receives a Packet Too Big message, it MUST reduce its estimate of the PMTU for the relevant path, based on the value of the MTU field in the message. The precise behavior of a node in this circumstance is not specified, since different applications may have different requirements, and since different implementation architectures may favor different strategies.

After receiving a Packet Too Big message, a node MUST attempt to avoid eliciting more such messages in the near future. The node MUST reduce the size of the packets it is sending along the path. Using a PMTU estimate larger than the IPv6 minimum link MTU may continue to elicit Packet Too Big messages. Since each of these messages (and the dropped packets they respond to) consume network resources, the node MUST force the Path MTU Discovery process to end.

Nodes using Path MTU Discovery MUST detect decreases in PMTU as fast as possible. Nodes MAY detect increases in PMTU, but because doing so requires sending packets larger than the current estimated PMTU,

and because the likelihood is that the PMTU will not have increased, this MUST be done at infrequent intervals. An attempt to detect an increase (by sending a packet larger than the current estimate) MUST NOT be done less than 5 minutes after a Packet Too Big message has been received for the given path. The recommended setting for this timer is twice its minimum value (10 minutes).

A node MUST NOT reduce its estimate of the Path MTU below the IPv6 minimum link MTU.

> Note: A node may receive a Packet Too Big message reporting a next-hop MTU that is less than the IPv6 minimum link MTU. In that case, the node is not required to reduce the size of subsequent packets sent on the path to less than the IPv6 minimun link MTU, but rather must include a Fragment header in those packets [IPv6-SPEC].

A node MUST NOT increase its estimate of the Path MTU in response to the contents of a Packet Too Big message. A message purporting to announce an increase in the Path MTU might be a stale packet that has been floating around in the network, a false packet injected as part of a denial-of-service attack, or the result of having multiple paths to the destination, each with a different PMTU.

5. Implementation Issues

This section discusses a number of issues related to the implementation of Path MTU Discovery. This is not a specification, but rather a set of notes provided as an aid for implementors.

The issues include:

- What layer or layers implement Path MTU Discovery?
- How is the PMTU information cached?
- How is stale PMTU information removed?
- What must transport and higher layers do?

5.1. Layering

In the IP architecture, the choice of what size packet to send is made by a protocol at a layer above IP. This memo refers to such a protocol as a "packetization protocol". Packetization protocols are usually transport protocols (for example, TCP) but can also be higher-layer protocols (for example, protocols built on top of UDP).

Implementing Path MTU Discovery in the packetization layers simplifies some of the inter-layer issues, but has several drawbacks: the implementation may have to be redone for each packetization protocol, it becomes hard to share PMTU information between different packetization layers, and the connection-oriented state maintained by some packetization layers may not easily extend to save PMTU information for long periods.

It is therefore suggested that the IP layer store PMTU information and that the ICMP layer process received Packet Too Big messages. The packetization layers may respond to changes in the PMTU, by changing the size of the messages they send. To support this layering, packetization layers require a way to learn of changes in the value of MMS_S, the "maximum send transport-message size". The MMS_S is derived from the Path MTU by subtracting the size of the IPv6 header plus space reserved by the IP layer for additional headers (if any).

It is possible that a packetization layer, perhaps a UDP application outside the kernel, is unable to change the size of messages it sends. This may result in a packet size that exceeds the Path MTU. To accommodate such situations, IPv6 defines a mechanism that allows large payloads to be divided into fragments, with each fragment sent in a separate packet (see [IPv6-SPEC] section "Fragment Header"). However, packetization layers are encouraged to avoid sending messages that will require fragmentation (for the case against fragmentation, see [FRAG]).

5.2. Storing PMTU information

Ideally, a PMTU value 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.

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

An implementation could use the destination address as the local representation of a path. The PMTU value associated with a destination would be the minimum PMTU learned across the set of all paths in use to that destination. The set of paths in use to a particular destination is expected to be small, in many cases consisting of a single path. This approach will result in the use of optimally sized packets on a per-destination basis. This approach integrates nicely with the conceptual model of a host as described in [ND]: a PMTU value could be stored with the corresponding entry in the destination cache.

If flows [IPv6-SPEC] are in use, an implementation could use the flow id 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 PMTU values maintained on a per-destination basis.

For source routed packets (i.e. packets containing an IPv6 Routing header [IPv6-SPEC]), the source route may further qualify the local representation of a path. In particular, a packet containing a type 0 Routing header in which all bits in the Strict/Loose Bit Map are equal to 1 contains a complete path specification. An implementation could use source route information in the local representation of a path.

> Note: Some paths may be further distinguished by different security classifications. The details of such classifications are beyond the scope of this memo.

Initially, the PMTU value for a path is assumed to be the (known) MTU of the first-hop link.

When a Packet Too Big message is received, the node determines which path the message applies to based on the contents of the Packet Too Big message. For example, if the destination address is used as the local representation of a path, the destination address from the original packet would be used to determine which path the message applies to.

> Note: if the original packet contained a Routing header, the Routing header should be used to determine the location of the destination address within the original packet. If Segments Left is equal to zero, the destination address is in the Destination Address field in the IPv6 header. If Segments Left is greater than zero, the destination address is the last address (Address[n]) in the Routing header.

The node then uses the value in the MTU field in the Packet Too Big message as a tentative PMTU value, and compares the tentative PMTU to the existing PMTU. If the tentative PMTU is less than the existing PMTU estimate, the tentative PMTU replaces the existing PMTU as the PMTU value for the path.

The packetization layers must be notified about decreases in the PMTU. Any packetization layer instance (for example, a TCP connection) that is actively using the path must be notified if the PMTU estimate is decreased.

> Note: even if the Packet Too Big message contains an Original Packet Header that refers to a UDP packet, the TCP layer must be notified if any of its connections use the given path.

Also, the instance that sent the packet that elicited the Packet Too Big message should be notified that its packet has been dropped, even if the PMTU estimate has not changed, so that it may retransmit the dropped data.

> Note: An implementation can avoid the use of an asynchronous notification mechanism for PMTU decreases by postponing notification until the next attempt to send a packet larger than the PMTU estimate. In this approach, when an attempt is made to SEND a packet that is larger than the PMTU estimate, the SEND function should fail and return a suitable error indication. This approach may be more suitable to a connectionless packetization layer (such as one using UDP), which (in some implementations) may be hard to "notify" from the ICMP layer. In this case, the normal timeout-based retransmission mechanisms would be used to recover from the dropped packets.

It is important to understand that the notification of the packetization layer instances using the path about the change in the PMTU is distinct from the notification of a specific instance that a packet has been dropped. The latter should be done as soon as practical (i.e., asynchronously from the point of view of the packetization layer instance), while the former may be delayed until a packetization layer instance wants to create a packet. Retransmission should be done for only for those packets that are known to be dropped, as indicated by a Packet Too Big message.

5.3. Purging stale PMTU information

Internetwork topology is dynamic; routes change over time. While the local representation of a path may remain constant, the actual path(s) in use may change. Thus, PMTU information cached by a node can become stale.

If the stale PMTU value is too large, this will be discovered almost immediately once a large enough packet is sent on the path. No such mechanism exists for realizing that a stale PMTU value is too small, so an implementation should "age" cached values. When a PMTU value has not been decreased for a while (on the order of 10 minutes), the PMTU estimate should be set to the MTU of the first-hop link, and the packetization layers should be notified of the change. This will cause the complete Path MTU Discovery process to take place again.

> Note: an implementation should provide a means for changing the timeout duration, including setting it to "infinity". For example, nodes attached to an FDDI link which is then attached to the rest of the Internet via a small MTU serial line are never going to discover a new non-local PMTU, so they should not have to put up with dropped packets every 10 minutes.

An upper layer must not retransmit data in response to an increase in the PMTU estimate, since this increase never comes in response to an indication of a dropped packet.

One approach to implementing PMTU aging is to associate a timestamp field with a PMTU value. This field is initialized to a "reserved" value, indicating that the PMTU is equal to the MTU of the first hop link. Whenever the PMTU is decreased in response to a Packet Too Big message, the timestamp is set to the current time.

Once a minute, a timer-driven procedure runs through all cached PMTU values, and for each PMTU whose timestamp is not "reserved" and is older than the timeout interval:

- The PMTU estimate is set to the MTU of the first hop link.

- The timestamp is set to the "reserved" value.

- Packetization layers using this path are notified of the increase.

5.4. TCP layer actions

The TCP layer must track the PMTU for the path(s) in use by a connection; it should not send segments that would result in packets larger than the PMTU. A simple implementation could ask the IP layer for this value each time it created a new segment, but this could be inefficient. Moreover, TCP implementations that follow the "slow-start" congestion-avoidance algorithm [CONG] typically calculate and cache several other values derived from the PMTU. It may be simpler to receive asynchronous notification when the PMTU changes, so that these variables may be updated.

A TCP implementation must also store the MSS value received from its peer, and must not send any segment larger than this MSS, regardless of the PMTU. In 4.xBSD-derived implementations, this may require adding an additional field to the TCP state record.

The value sent in the TCP MSS option is independent of the PMTU. This MSS option value is used by the other end of the connection, which may be using an unrelated PMTU value. See [IPv6-SPEC] sections "Packet Size Issues" and "Maximum Upper-Layer Payload Size" for information on selecting a value for the TCP MSS option.

When a Packet Too Big message is received, it implies that a packet was dropped by the node that sent the ICMP message. It is sufficient to treat this as any other dropped segment, and wait until the retransmission timer expires to cause retransmission of the segment. If the Path MTU Discovery process requires several steps to find the PMTU of the full path, this could delay the connection by many round-trip times.

Alternatively, the retransmission could be done in immediate response to a notification that the Path MTU has changed, but only for the specific connection specified by the Packet Too Big message. The packet size used in the retransmission should be no larger than the new PMTU.

> Note: A packetization layer must not retransmit in response to every Packet Too Big message, since a burst of several oversized segments will give rise to several such messages and hence several retransmissions of the same data. If the new estimated PMTU is still wrong, the process repeats, and there is an exponential growth in the number of superfluous segments sent.
>
> This means that the TCP layer must be able to recognize when a Packet Too Big notification actually decreases the PMTU that it has already used to send a packet on the given connection, and should ignore any other notifications.

Many TCP implementations incorporate "congestion avoidance" and "slow-start" algorithms to improve performance [CONG]. Unlike a retransmission caused by a TCP retransmission timeout, a retransmission caused by a Packet Too Big message should not change the congestion window. It should, however, trigger the slow-start mechanism (i.e., only one segment should be retransmitted until acknowledgements begin to arrive again).

TCP performance can be reduced if the sender’s maximum window size is not an exact multiple of the segment size in use (this is not the congestion window size, which is always a multiple of the segment

size). In many systems (such as those derived from 4.2BSD), the segment size is often set to 1024 octets, and the maximum window size (the "send space") is usually a multiple of 1024 octets, so the proper relationship holds by default. If Path MTU Discovery is used, however, the segment size may not be a submultiple of the send space, and it may change during a connection; this means that the TCP layer may need to change the transmission window size when Path MTU Discovery changes the PMTU value. The maximum window size should be set to the greatest multiple of the segment size that is less than or equal to the sender's buffer space size.

5.5. Issues for other transport protocols

Some transport protocols (such as ISO TP4 [ISOTP]) are not allowed to repacketize 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. Subsequent segments, when transmitted for the first time, should be no larger than allowed by the Path MTU.

The Sun Network File System (NFS) uses a Remote Procedure Call (RPC) protocol [RPC] that, when used over UDP, in many cases will generate payloads that must be fragmented even for the first-hop link. This might improve performance in certain cases, but it is known to cause reliability and performance problems, especially when the client and server are separated by routers.

It is recommended that NFS implementations use Path MTU Discovery whenever routers are involved. Most NFS implementations allow the RPC datagram size to be changed at mount-time (indirectly, by changing the effective file system block size), but might require some modification to support changes later on.

Also, since a single NFS operation cannot be split across several UDP datagrams, certain operations (primarily, those operating on file names and directories) require a minimum payload size that if sent in a single packet would exceed the PMTU. NFS implementations should not reduce the payload size below this threshold, even if Path MTU Discovery suggests a lower value. In this case the payload will be fragmented by the IP layer.

5.6. Management interface

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

- Specify that Path MTU Discovery not be done on a given path.

- Change the PMTU value associated with a given path.

The former can be accomplished by associating a flag with the path; when a packet is sent on a path with this flag set, the IP layer does not send packets larger than the IPv6 minimum link MTU.

These features might be used to work around an anomalous situation, or by a routing protocol implementation that is able to obtain Path MTU values.

The implementation should also provide a way to change the timeout period for aging stale PMTU information.

6. Security Considerations

This Path MTU Discovery mechanism makes possible two denial-of-service attacks, both based on a malicious party sending false Packet Too Big messages to a node.

In the first attack, the false message indicates a PMTU much smaller than reality. This should not entirely stop data flow, since the victim node should never set its PMTU estimate below the IPv6 minimum link MTU. It will, however, result in suboptimal performance.

In the second attack, the false message indicates a PMTU larger than reality. If believed, this could cause temporary blockage as the victim sends packets that will be dropped by some router. Within one round-trip time, the node would discover its mistake (receiving Packet Too Big messages from that router), but frequent repetition of this attack could cause lots of packets to be dropped. A node, however, should never raise its estimate of the PMTU based on a Packet Too Big message, so should not be vulnerable to this attack.

A malicious party could also cause problems if it could stop a victim from receiving legitimate Packet Too Big messages, but in this case there are simpler denial-of-service attacks available.

Acknowledgements

We would like to acknowledge the authors of and contributors to [RFC-1191], from which the majority of this document was derived. We would also like to acknowledge the members of the IPng working group for their careful review and constructive criticisms.

Appendix A - Comparison to RFC 1191

This document is based in large part on RFC 1191, which describes Path MTU Discovery for IPv4. Certain portions of RFC 1191 were not needed in this document:

router specification	- Packet Too Big messages and corresponding router behavior are defined in [ICMPv6]
Don't Fragment bit	- there is no DF bit in IPv6 packets
TCP MSS discussion	- selecting a value to send in the TCP MSS option is discussed in [IPv6-SPEC]
old-style messages	- all Packet Too Big messages report the MTU of the constricting link
MTU plateau tables	- not needed because there are no old-style messages

References

[CONG] Van Jacobson. Congestion Avoidance and Control. Proc. SIGCOMM '88 Symposium on Communications Architectures and Protocols, pages 314-329. Stanford, CA, August, 1988.

[FRAG] C. Kent and J. Mogul. Fragmentation Considered Harmful. In Proc. SIGCOMM '87 Workshop on Frontiers in Computer Communications Technology. August, 1987.

[ICMPv6] Conta, A., and S. Deering, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 1885, December 1995.

[IPv6-SPEC] Deering, S., and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 1883, December 1995.

[ISOTP] ISO. ISO Transport Protocol Specification: ISO DP 8073. RFC 905, SRI Network Information Center, April, 1984.

[ND] Narten, T., Nordmark, E., and W. Simpson, "Neighbor Discovery for IP Version 6 (IPv6)", Work in Progress.

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

[RPC] Sun Microsystems, Inc., "RPC: Remote Procedure Call Protocol", RFC 1057, SRI Network Information Center, June, 1988.

Authors' Addresses

Jack McCann
Digital Equipment Corporation
110 Spitbrook Road, ZKO3-3/U14
Nashua, NH 03062
Phone: +1 603 881 2608

Fax: +1 603 881 0120
Email: mccann@zk3.dec.com

Stephen E. Deering
Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304
Phone: +1 415 812 4839

Fax: +1 415 812 4471
EMail: deering@parc.xerox.com

Jeffrey Mogul
Digital Equipment Corporation Western Research Laboratory
250 University Avenue
Palo Alto, CA 94301
Phone: +1 415 617 3304

EMail: mogul@pa.dec.com