                          Path MTU Discovery

Status of this Memo

   This RFC specifies a protocol on the IAB Standards Track for the
   Internet community, and requests discussion and suggestions for
   improvements. Please refer to the current edition of the "IAB
   Official Protocol Standards" for the standardization state and status
   of this protocol. Distribution of this memo is unlimited.

                          Table of Contents

                          List of Tables

Abstract

   This memo describes a technique for dynamically discovering the
   maximum transmission unit (MTU) of an arbitrary internet path. It
   specifies a small change to the way routers generate one type of ICMP
   message. For a path that passes through a router that has not been
   so changed, this technique might not discover the correct Path MTU,
   but it will always choose a Path MTU as accurate as, and in many
   cases more accurate than, the Path MTU that would be chosen by
   current practice.

Acknowledgements

   This proposal is a product of the IETF MTU Discovery Working Group.

   The mechanism proposed here was first suggested by Geof Cooper [2],
   who in two short paragraphs set out all the basic ideas that took the
   Working Group months to reinvent.

1. Introduction

   When one IP host has a large amount of data to send to another host,
   the data is transmitted as a series of IP datagrams. It is usually
   preferable that these datagrams be of the largest size that does not
   require fragmentation anywhere along the path from the source to the
   destination. (For the case against fragmentation, see [5].) This
   datagram size is referred to as the Path MTU (PMTU), and it is equal
   to the minimum of the MTUs of each hop in the path. A shortcoming of
   the current Internet protocol suite is the lack of a standard
   mechanism for a host to discover the PMTU of an arbitrary path.

        Note: The Path MTU is what in [1] is called the "Effective MTU
        for sending" (EMTU_S). A PMTU is associated with a path,
        which is a particular combination of IP source and destination
        address and perhaps a Type-of-service (TOS).

   The current practice [1] is to use the lesser of 576 and the
   first-hop MTU as the PMTU for any destination that is not connected
   to the same network or subnet as the source. In many cases, this
   results in the use of smaller datagrams than necessary, because many
   paths have a PMTU greater than 576. A host sending datagrams much
   smaller than the Path MTU allows is wasting Internet resources and
   probably getting suboptimal throughput. Furthermore, current
   practice does not prevent fragmentation in all cases, since there are
   some paths whose PMTU is less than 576.

It is expected that future routing protocols will be able to provide
accurate PMTU information within a routing area, although perhaps not
across multi-level routing hierarchies. It is not clear how soon
that will be ubiquitously available, so for the next several years
the Internet needs a simple mechanism that discovers PMTUs without
wasting resources and that works before all hosts and routers are
modified.

2. Protocol overview

In this memo, we describe a technique for using the Don't Fragment
(DF) bit in the IP header to dynamically discover the PMTU of a path.
The basic idea is that a source host initially assumes that the PMTU
of a path is the (known) MTU of its first hop, and sends all
datagrams on that path with the DF bit set. If any of the datagrams
are too large to be forwarded without fragmentation by some router
along the path, that router will discard them and return ICMP
Destination Unreachable messages with a code meaning "fragmentation
needed and DF set" [7]. Upon receipt of such a message (henceforth
called a "Datagram Too Big" message), the source host reduces its
assumed PMTU for the path.

The PMTU discovery process ends when the host's estimate of the PMTU
is low enough that its datagrams can be delivered without
fragmentation. Or, the host may elect to end the discovery process
by ceasing to set the DF bit in the datagram headers; it may do so,
for example, because it is willing to have datagrams fragmented in
some circumstances. Normally, the host continues to set DF in all
datagrams, so that if the route changes and the new PMTU is lower, it
will be discovered.

Unfortunately, the Datagram Too Big message, as currently specified,
does not report the MTU of the hop for which the rejected datagram
was too big, so the source host cannot tell exactly how much to
reduce its assumed PMTU. To remedy this, we propose that a currently
unused header field in the Datagram Too Big message be used to report
the MTU of the constricting hop. This is the only change specified
for routers in support of PMTU Discovery.

The PMTU of a path may change over time, due to changes in the
routing topology. Reductions of the PMTU are detected by Datagram
Too Big messages, except on paths for which the host has stopped
setting the DF bit. To detect increases in a path's PMTU, a host
periodically increases its assumed PMTU (and if it had stopped,
resumes setting the DF bit). This will almost always result in
datagrams being discarded and Datagram Too Big messages being

generated, because in most cases the PMTU of the path will not have
changed, so it should be done infrequently.

Since this mechanism essentially guarantees that host will not
receive any fragments from a peer doing PMTU Discovery, it may aid in
interoperating with certain hosts that (improperly) are unable to
reassemble fragmented datagrams.

3. Host specification

When a host receives a Datagram Too Big message, it MUST reduce its
estimate of the PMTU for the relevant path, based on the value of the
Next-Hop MTU field in the message (see section 4). We do not specify
the precise behavior of a host in this circumstance, since different
applications may have different requirements, and since different
implementation architectures may favor different strategies.

We do require that after receiving a Datagram Too Big message, a host
MUST attempt to avoid eliciting more such messages in the near
future. The host may either reduce the size of the datagrams it is
sending along the path, or cease setting the Don't Fragment bit in
the headers of those datagrams. Clearly, the former strategy may
continue to elicit Datagram Too Big messages for a while, but since
each of these messages (and the dropped datagrams they respond to)
consume Internet resources, the host MUST force the PMTU Discovery
process to converge.

Hosts using PMTU Discovery MUST detect decreases in Path MTU as fast
as possible. Hosts MAY detect increases in Path MTU, but because
doing so requires sending datagrams 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 datagram larger than the current
estimate) MUST NOT be done less than 5 minutes after a Datagram Too
Big message has been received for the given destination, or less than
1 minute after a previous, successful attempted increase. We
recommend setting these timers at twice their minimum values (10
minutes and 2 minutes, respectively).

Hosts MUST be able to deal with Datagram Too Big messages that do not
include the next-hop MTU, since it is not feasible to upgrade all the
routers in the Internet in any finite time. A Datagram Too Big
message from an unmodified router can be recognized by the presence
of a zero in the (newly-defined) Next-Hop MTU field. (This is
required by the ICMP specification [7], which says that "unused"
fields must be zero.) In section 5, we discuss possible strategies

for a host to follow in response to an old-style Datagram Too Big
message (one sent by an unmodified router).

A host MUST never reduce its estimate of the Path MTU below 68
octets.

A host MUST not increase its estimate of the Path MTU in response to
the contents of a Datagram Too Big message. A message purporting to
announce an increase in the Path MTU might be a stale datagram that
has been floating around in the Internet, a false packet injected as
part of a denial-of-service attack, or the result of having multiple
paths to the destination.

3.1. TCP MSS Option

A host doing PMTU Discovery must obey the rule that it not send IP
datagrams larger than 576 octets unless it has permission from the
receiver. For TCP connections, this means that a host must not send
datagrams larger than 40 octets plus the Maximum Segment Size (MSS)
sent by its peer.

> Note: The TCP MSS is defined to be the relevant IP datagram
> size minus 40 [9]. The default of 576 octets for the maximum
> IP datagram size yields a default of 536 octets for the TCP
> MSS.

Section 4.2.2.6 of "Requirements for Internet Hosts -- Communication
Layers" [1] says:

> Some TCP implementations send an MSS option only if the
> destination host is on a non-connected network. However, in
> general the TCP layer may not have the appropriate information
> to make this decision, so it is preferable to leave to the IP
> layer the task of determining a suitable MTU for the Internet
> path.

Actually, many TCP implementations always send an MSS option, but set
the value to 536 if the destination is non-local. This behavior was
correct when the Internet was full of hosts that did not follow the
rule that datagrams larger than 576 octets should not be sent to
non-local destinations. Now that most hosts do follow this rule, it
is unnecessary to limit the value in the TCP MSS option to 536 for
non-local peers.

Moreover, doing this prevents PMTU Discovery from discovering PMTUs
larger than 576, so hosts SHOULD no longer lower the value they send

in the MSS option. The MSS option should be 40 octets less than the
size of the largest datagram the host is able to reassemble (MMS_R,
as defined in [1]); in many cases, this will be the architectural
limit of 65495 (65535 - 40) octets. A host MAY send an MSS value
derived from the MTU of its connected network (the maximum MTU over
its connected networks, for a multi-homed host); this should not
cause problems for PMTU Discovery, and may dissuade a broken peer
from sending enormous datagrams.

> Note: At the moment, we see no reason to send an MSS greater
> than the maximum MTU of the connected networks, and we
> recommend that hosts do not use 65495. It is quite possible
> that some IP implementations have sign-bit bugs that would be
> tickled by unnecessary use of such a large MSS.

4. Router specification

When a router is unable to forward a datagram because it exceeds the
MTU of the next-hop network and its Don't Fragment bit is set, the
router is required to return an ICMP Destination Unreachable message
to the source of the datagram, with the Code indicating
"fragmentation needed and DF set". To support the Path MTU Discovery
technique specified in this memo, the router MUST include the MTU of
that next-hop network in the low-order 16 bits of the ICMP header
field that is labelled "unused" in the ICMP specification [7]. The
high-order 16 bits remain unused, and MUST be set to zero. Thus, the
message has the following format:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   Type = 3    |   Code = 4    |           Checksum            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |           unused = 0          |         Next-Hop MTU          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      Internet Header + 64 bits of Original Datagram Data      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The value carried in the Next-Hop MTU field is:

> The size in octets of the largest datagram that could be
> forwarded, along the path of the original datagram, without
> being fragmented at this router. The size includes the IP
> header and IP data, and does not include any lower-level
> headers.

This field will never contain a value less than 68, since every
router "must be able to forward a datagram of 68 octets without
fragmentation" [8].

5. Host processing of old-style messages

   In this section we outline several possible strategies for a host to
   follow upon receiving a Datagram Too Big message from an unmodified
   router (i.e., one where the Next-Hop MTU field is zero). This
   section is not part of the protocol specification.

   The simplest thing for a host to do in response to such a message is
   to assume that the PMTU is the minimum of its currently-assumed PMTU
   and 576, and to stop setting the DF bit in datagrams sent on that
   path. Thus, the host falls back to the same PMTU as it would choose
   under current practice (see section 3.3.3 of "Requirements for
   Internet Hosts -- Communication Layers" [1]). This strategy has the
   advantage that it terminates quickly, and does no worse than existing
   practice. It fails, however, to avoid fragmentation in some cases,
   and to make the most efficient utilization of the internetwork in
   other cases.

   More sophisticated strategies involve "searching" for an accurate
   PMTU estimate, by continuing to send datagrams with the DF bit while
   varying their sizes. A good search strategy is one that obtains an
   accurate estimate of the Path MTU without causing many packets to be
   lost in the process.

   Several possible strategies apply algorithmic functions to the
   previous PMTU estimate to generate a new estimate. For example, one
   could multiply the old estimate by a constant (say, 0.75). We do NOT
   recommend this; it either converges far too slowly, or it
   substantially underestimates the true PMTU.

   A more sophisticated approach is to do a binary search on the packet
   size. This converges somewhat faster, although it still takes 4 or 5
   steps to converge from an FDDI MTU to an Ethernet MTU. A serious
   disadvantage is that it requires a complex implementation in order to
   recognize when a datagram has made it to the other end (indicating
   that the current estimate is too low). We also do not recommend this
   strategy.

   One strategy that appears to work quite well starts from the
   observation that there are, in practice, relatively few MTU values in
   use in the Internet. Thus, rather than blindly searching through
   arbitrarily chosen values, we can search only the ones that are

likely to appear. Moreover, since designers tend to chose MTUs in
similar ways, it is possible to collect groups of similar MTU values
and use the lowest value in the group as our search "plateau". (It
is clearly better to underestimate an MTU by a few per cent than to
overestimate it by one octet.)

In section 7, we describe how we arrived at a table of representative
MTU plateaus for use in PMTU estimation. With this table,
convergence is as good as binary search in the worst case, and is far
better in common cases (for example, it takes only two round-trip
times to go from an FDDI MTU to an Ethernet MTU). Since the plateaus
lie near powers of two, if an MTU is not represented in this table,
the algorithm will not underestimate it by more than a factor of 2.

Any search strategy must have some "memory" of previous estimates in
order to chose the next one. One approach is to use the
currently-cached estimate of the Path MTU, but in fact there is
better information available in the Datagram Too Big message itself.
All ICMP Destination Unreachable messages, including this one,
contain the IP header of the original datagram, which contains the
Total Length of the datagram that was too big to be forwarded without
fragmentation. Since this Total Length may be less than the current
PMTU estimate, but is nonetheless larger than the actual PMTU, it may
be a good input to the method for choosing the next PMTU estimate.

> Note: routers based on implementations derived from 4.2BSD
> Unix send an incorrect value for the Total Length of the
> original IP datagram. The value sent by these routers is the
> sum of the original Total Length and the original Header
> Length (expressed in octets). Since it is impossible for the
> host receiving such a Datagram Too Big message to know if it
> sent by one of these routers, the host must be conservative
> and assume that it is. If the Total Length field returned is
> not less than the current PMTU estimate, it must be reduced by
> 4 times the value of the returned Header Length field.

The strategy we recommend, then, is to use as the next PMTU estimate
the greatest plateau value that is less than the returned Total
Length field (corrected, if necessary, according to the Note above).

6. Host implementation

In this section we discuss how PMTU Discovery is implemented in host
software. This is not a specification, but rather a set of
suggestions.

The issues include:

- What layer or layers implement PMTU Discovery?

- Where is the PMTU information cached?

- How is stale PMTU information removed?

- What must transport and higher layers do?

6.1. Layering

   In the IP architecture, the choice of what size datagram to send is
   made by a protocol at a layer above IP. We refer 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 PMTU 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.

   We therefore believe that the IP layer should store PMTU information
   and that the ICMP layer should process received Datagram Too Big
   messages. The packetization layers must still be able to respond to
   changes in the Path MTU, by changing the size of the datagrams they
   send, and must also be able to specify that datagrams are sent with
   the DF bit set. We do not want the IP layer to simply set the DF bit
   in every packet, since it is possible that a packetization layer,
   perhaps a UDP application outside the kernel, is unable to change its
   datagram size. Protocols involving intentional fragmentation, while
   inelegant, are sometimes successful (NFS being the primary example),
   and we do not want to break such protocols.

   To support this layering, packetization layers require an extension
   of the IP service interface defined in [1]:

        A way to learn of changes in the value of MMS_S, the "maximum
        send transport-message size", which is derived from the Path
        MTU by subtracting the minimum IP header size.

6.2. Storing PMTU information

   In general, the IP layer should associate each PMTU value that it has
   learned with a specific path. A path is identified by a source
   address, a destination address and an IP type-of-service. (Some
   implementations do not record the source address of paths; this is
   acceptable for single-homed hosts, which have only one possible
   source address.)

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

   The obvious place to store this association is as a field in the
   routing table entries. A host will not have a route for every
   possible destination, but it should be able to cache a per-host route
   for every active destination. (This requirement is already imposed
   by the need to process ICMP Redirect messages.)

   When the first packet is sent to a host for which no per-host route
   exists, a route is chosen either from the set of per-network routes,
   or from the set of default routes. The PMTU fields in these route
   entries should be initialized to be the MTU of the associated
   first-hop data link, and must never be changed by the PMTU Discovery
   process. (PMTU Discovery only creates or changes entries for
   per-host routes). Until a Datagram Too Big message is received, the
   PMTU associated with the initially-chosen route is presumed to be
   accurate.

   When a Datagram Too Big message is received, the ICMP layer
   determines a new estimate for the Path MTU (either from a non-zero
   Next-Hop MTU value in the packet, or using the method described in
   section 5). If a per-host route for this path does not exist, then
   one is created (almost as if a per-host ICMP Redirect is being
   processed; the new route uses the same first-hop router as the
   current route). If the PMTU estimate associated with the per-host
   route is higher than the new estimate, then the value in the routing
   entry is changed.

   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 Datagram Too Big message contains an
        Original Datagram 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 datagram that elicited the Datagram
Too Big message should be notified that its datagram has been
dropped, even if the PMTU estimate has not changed, so that it may
retransmit the dropped datagram.

Note: The notification mechanism can be analogous to the
mechanism used to provide notification of an ICMP Source
Quench message. In some implementations (such as
4.2BSD-derived systems), the existing notification mechanism
is not able to identify the specific connection involved, and
so an additional mechanism is necessary.

Alternatively, an implementation can avoid the use of an
asynchronous notification mechanism for PMTU decreases by
postponing notification until the next attempt to send a
datagram larger than the PMTU estimate. In this approach,
when an attempt is made to SEND a datagram with the DF bit
set, and the datagram 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 datagrams.

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 Datagram Too Big message.

6.3. Purging stale PMTU information

Internetwork topology is dynamic; routes change over time. The PMTU
discovered for a given destination may be wrong if a new route comes
into use. Thus, PMTU information cached by a host can become stale.

Because a host using PMTU Discovery always sets the DF bit, if the
stale PMTU value is too large, this will be discovered almost

immediately once a datagram is sent to the given destination. 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 first-hop data-link
MTU, and the packetization layers should be notified of the change.
This will cause the complete PMTU 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, hosts attached to an FDDI network which is then
     attached to the rest of the Internet via a slow serial line
     are never going to discover a new non-local PMTU, so they
     should not have to put up with dropped datagrams every 10
     minutes.

An upper layer MUST not retransmit datagrams in response to an
increase in the PMTU estimate, since this increase never comes in
response to an indication of a dropped datagram.

One approach to implementing PMTU aging is to add a timestamp field
to the routing table entry. This field is initialized to a
"reserved" value, indicating that the PMTU has never been changed.
Whenever the PMTU is decreased in response to a Datagram Too Big
message, the timestamp is set to the current time.

Once a minute, a timer-driven procedure runs through the routing
table, and for each entry whose timestamp is not "reserved" and is
older than the timeout interval:

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

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

PMTU estimates may disappear from the routing table if the per-host
routes are removed; this can happen in response to an ICMP Redirect
message, or because certain routing-table daemons delete old routes
after several minutes. Also, on a multi-homed host a topology change
may result in the use of a different source interface. When this
happens, if the packetization layer is not notified then it may
continue to use a cached PMTU value that is now too small. One
solution is to notify the packetization layer of a possible PMTU
change whenever a Redirect message causes a route change, and
whenever a route is simply deleted from the routing table.

         Note: a more sophisticated method for detecting PMTU increases
         is described in section 7.1.

6.4. TCP layer actions

   The TCP layer must track the PMTU for the destination of a
   connection; it should not send datagrams that would be larger than
   this. A simple implementation could ask the IP layer for this value
   (using the GET_MAXSIZES interface described in [1]) each time it
   created a new segment, but this could be inefficient. Moreover, TCP
   implementations that follow the "slow-start" congestion-avoidance
   algorithm [4] 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 (which defaults to 536), and not send any segment larger than
   this MSS, regardless of the PMTU. In 4.xBSD-derived implementations,
   this requires adding an additional field to the TCP state record.

   Finally, when a Datagram Too Big message is received, it implies that
   a datagram was dropped by the router 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 PMTU Discovery process requires several steps to
   estimate the right PMTU, 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 Datagram Too Big message. The
   datagram size used in the retransmission should, of course, be no
   larger than the new PMTU.

         Note: One MUST not retransmit in response to every Datagram
         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
         Datagram Too Big notification actually decreases the PMTU that
         it has already used to send a datagram on the given
         connection, and should ignore any other notifications.

Modern TCP implementations incorporate "congestion advoidance" and
"slow-start" algorithms to improve performance [4]. Unlike a
retransmission caused by a TCP retransmission timeout, a
retransmission caused by a Datagram 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 system (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 PMTU 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 PMTU Discovery
changes the PMTU value. The maximum window size should be set to the
greatest multiple of the segment size (PMTU - 40) that is less than
or equal to the sender's buffer space size.

PMTU Discovery does not affect the value sent in the TCP MSS option,
because that value is used by the other end of the connection, which
may be using an unrelated PMTU value.

6.5. Issues for other transport protocols

Some transport protocols (such as ISO TP4 [3]) are not allowed to
repacketize when doing a retransmission. That is, once an attempt is
made to transmit a datagram of a certain size, its contents cannot be
split into smaller datagrams for retransmission. In such a case, the
original datagram should be retransmitted without the DF bit set,
allowing it to be fragmented as necessary to reach its destination.
Subsequent datagrams, when transmitted for the first time, should be
no larger than allowed by the Path MTU, and should have the DF bit
set.

The Sun Network File System (NFS) uses a Remote Procedure Call (RPC)
protocol [11] that, in many cases, sends datagrams 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.

We recommend that NFS implementations use PMTU 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 datagram size that may be
larger than the PMTU. NFS implementations should not reduce the
datagram size below this threshold, even if PMTU Discovery suggests a
lower value. (Of course, in this case datagrams should not be sent
with DF set.)

6.6. Management interface

We suggest that an implementation provide a way for a system utility
program to:

   - Specify that PMTU Discovery not be done on a given route.

   - Change the PMTU value associated with a given route.

The former can be accomplished by associating a flag with the routing
entry; when a packet is sent via a route with this flag set, the IP
layer leaves the DF bit clear no matter what the upper layer
requests.

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.

7. Likely values for Path MTUs

The algorithm recommended in section 5 for "searching" the space of
Path MTUs is based on a table of values that severely restricts the
search space. We describe here a table of MTU values that, as of
this writing, represents all major data-link technologies in use in
the Internet.

In table 7-1, data links are listed in order of decreasing MTU, and
grouped so that each set of similar MTUs is associated with a
"plateau" equal to the lowest MTU in the group. (The table also

includes some entries not currently associated with a data link, and
gives references where available). Where a plateau represents more
than one MTU, the table shows the maximum inaccuracy associated with
the plateau, as a percentage.

We do not expect that the values in the table, especially for higher
MTU levels, are going to be valid forever. The values given here are
an implementation suggestion, NOT a specification or requirement.
Implementors should use up-to-date references to pick a set of
plateaus; it is important that the table not contain too many entries
or the process of searching for a PMTU might waste Internet
resources. Implementors should also make it convenient for customers
without source code to update the table values in their systems (for
example, the table in a BSD-derived Unix kernel could be changed
using a new "ioctl" command).

> Note: It might be a good idea to add a few table entries for
> values equal to small powers of 2 plus 40 (for the IP and TCP
> headers), where no similar values exist, since this seems to
> be a reasonably non-arbitrary way of choosing arbitrary
> values.
>
> The table might also contain entries for values slightly less
> than large powers of 2, in case MTUs are defined near those
> values (it is better in this case for the table entries to be
> low than to be high, or else the next lowest plateau may be
> chosen instead).

7.1. A better way to detect PMTU increases

Section 6.3 suggests detecting increases in the PMTU value by
periodically increasing the PTMU estimate to the first-hop MTU.
Since it is likely that this process will simply "rediscover" the
current PTMU estimate, at the cost of several dropped datagrams, it
should not be done often.

A better approach is to periodically increase the PMTU estimate to
the next-highest value in the plateau table (or the first-hop MTU, if
that is smaller). If the increased estimate is wrong, at most one
round-trip time is wasted before the correct value is rediscovered.
If the increased estimate is still too low, a higher estimate will be
attempted somewhat later.

Because it may take several such periods to discover a significant
increase in the PMTU, we recommend that a short timeout period should
be used after the estimate is increased, and a longer timeout be used

Plateau	MTU	Comments	Reference
	65535	Official maximum MTU	RFC 791
	65535	Hyperchannel	RFC 1044
65535			
32000		Just in case	
	17914	16Mb IBM Token Ring	ref. [6]
17914			
	8166	IEEE 802.4	RFC 1042
8166			
	4464	IEEE 802.5 (4Mb max)	RFC 1042
	4352	FDDI (Revised)	RFC 1188
4352 (1%)			
	2048	Wideband Network	RFC 907
	2002	IEEE 802.5 (4Mb recommended)	RFC 1042
2002 (2%)			
	1536	Exp. Ethernet Nets	RFC 895
	1500	Ethernet Networks	RFC 894
	1500	Point-to-Point (default)	RFC 1134
	1492	IEEE 802.3	RFC 1042
1492 (3%)			
	1006	SLIP	RFC 1055
	1006	ARPANET	BBN 1822
1006			
	576	X.25 Networks	RFC 877
	544	DEC IP Portal	ref. [10]
	512	NETBIOS	RFC 1088
	508	IEEE 802/Source-Rt Bridge	RFC 1042
	508	ARCNET	RFC 1051
508 (13%)			
	296	Point-to-Point (low delay)	RFC 1144
296			
68		Official minimum MTU	RFC 791

Table 7-1: Common MTUs in the Internet

after the PTMU estimate is decreased because of a Datagram Too Big
message. For example, after the PTMU estimate is decreased, the
timeout should be set to 10 minutes; once this timer expires and a
larger MTU is attempted, the timeout can be set to a much smaller
value (say, 2 minutes). In no case should the timeout be shorter
than the estimated round-trip time, if this is known.

8. Security considerations

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

   In the first attack, the false message indicates a PMTU much smaller
   than reality. This should not entirely stop data flow, since the
   victim host should never set its PMTU estimate below the absolute
   minimum, but at 8 octets of IP data per datagram, progress could be
   slow.

   In the other attack, the false message indicates a PMTU greater than
   reality. If believed, this could cause temporary blockage as the
   victim sends datagrams that will be dropped by some router. Within
   one round-trip time, the host would discover its mistake (receiving
   Datagram Too Big messages from that router), but frequent repetition
   of this attack could cause lots of datagrams to be dropped. A host,
   however, should never raise its estimate of the PMTU based on a
   Datagram 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 Datagram Too Big messages, but in this case
   there are simpler denial-of-service attacks available.

References

[1]   R. Braden, ed. Requirements for Internet Hosts -- Communication
      Layers. RFC 1122, SRI Network Information Center, October, 1989.

[2]   Geof Cooper. IP Datagram Sizes. Electronic distribution of the
      TCP-IP Discussion Group, Message-ID
      <8705240517.AA01407@apolling.imagen.uucp>.

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

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

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

[6]   Drew Daniel Perkins. Private Communication.

[7]   J. Postel. Internet Control Message Protocol. RFC 792, SRI
      Network Information Center, September, 1981.

[8]   J. Postel. Internet Protocol. RFC 791, SRI Network Information
      Center, September, 1981.

[9]   J. Postel. The TCP Maximum Segment Size and Related Topics. RFC
      879, SRI Network Information Center, November, 1983.

[10] Michael Reilly. Private Communication.

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

Authors' Addresses

   Jeffrey Mogul
   Digital Equipment Corporation Western Research Laboratory
   100 Hamilton Avenue
   Palo Alto, CA 94301

   Phone: (415) 853-6643
   EMail: mogul@decwrl.dec.com

   Steve Deering
   Xerox Palo Alto Research Center
   3333 Coyote Hill Road
   Palo Alto, CA 94304

   Phone: (415) 494-4839
   EMail: deering@xerox.com