Network Working Group
Internet-Draft
Intended status: Experimental
Expires: September 22, 2016

I. van Beijnum
Institute IMDEA Networks

March 21, 2016

Extensions for Multi-MTU Subnets
draft-van-beijnum-multi-mtu-05

Abstract

In the early days of the internet, many different link types with many different maximum packet sizes were in use. For point-to-point or point-to-multipoint links, there are still some other link types (PPP, ATM, Packet over SONET), but multipoint subnets are now almost exclusively implemented as Ethernets. Even though the relevant standards mandate a 1500 byte maximum packet size for Ethernet, more and more Ethernet equipment is capable of handling packets bigger than 1500 bytes. However, since this capability isn't standardized, it is seldom used today, despite the potential performance benefits of using larger packets. This document specifies mechanisms to negotiate per-neighbor maximum packet sizes so that nodes on a multipoint subnet may use the maximum mutually supported packet size between them without being limited by nodes with smaller maximum sizes on the same subnet.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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

This Internet-Draft will expire on September 22, 2016.

Table of Contents

1. Introduction

Some protocols inherently generate small packets. Examples are VoIP, where it is necessary to send packets frequently before much data can be gathered to fill up the packet, and the DNS, where the queries are inherently small and the returned results also often do not fill up a full 1500-byte packet. However, most data that is transferred across the internet and private networks is part of long-lived sessons and requires segmentation by a transport protocol, which is almost always TCP. These types of data transfers can benefit from larger packets in several ways:

1. A higher data-to-header ratio makes for fewer overhead bytes

2. Fewer packets means fewer per-packet operations for the source and destination hosts

3. Fewer packets also means fewer per-packet operations in routers and middleboxes

4. TCP performance increases with larger packet sizes

Even though today, the capability to use larger packets (often called jumboframes) is present in a lot of Ethernet hardware, this capability typically isn't used because IP assumes a common MTU size for all nodes connected to a link or subnet. In practice, this means that using a larger MTU requires manual configuration of the non-standard MTU size on all hosts and routers and possibly on layer 2 switches connected to a subnet. Also, the MTU size for a subnet is limited to that of the least capable router, host or switch.

Perhaps in the future, when hosts support packetization layer path MTU discovery ([RFC4821], "Packetization Layer Path MTU Discovery") in all relevant transport protocols, it will be possible to simply ignore MTU limitations by sending at the maximum locally supported size and determining the maximum packet size towards a correspondent from acknowledgements that come back for packets of different sizes. However, [RFC4821] must be implemented in every transport protocol, and problems arise in the case where hosts implementing [RFC4821] interact with hosts that don't implement this mechanism, but do use a larger than standard MTU.

This document provides for a set of mechanisms that allow the use of larger packets between nodes that support them which interacts well with both manually configured non-standard MTUs and expected future [RFC4821] operation with larger MTUs. This is done using a new IPv6 Neighbor Discovery option and a new UDP-based protocol for exchanging

MTU information and testing whether jumboframes can be transmitted successfully.

Appendix B discusses several potential issues with larger packets, such as head-of-line blocking delays, path MTU discovery black holes and the strength of the CRC32 with increasing packet sizes.

2. Notational Conventions

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].

Note that this specification is not standards track, and as such, can't overrule existing specifications. Whenever [RFC2119] language is used, this must be interpreted within the context of this specification: while the specification as a whole is optional and non-standard, whenever it is implemented, such an implementation can only function properly when all MUSTs are observed.

3. Terminology

Advertised MTU: The MTU size announced by a node to other nodes on the local subnet.

Confirmed MTU: The largest packet size successfully received from the neighbor or the largest packet size sent to the neighbor for which an acknowledgment was received; whichever size is greater.

Confirmed Time: When a packet the size of the confirmed MTU was last received or acknowledged.

Local MTU: The MTU configured on an interface. By default, this is the largest MTU size supported by the hardware, but the Local MTU may be lowered administratively or automatically based on policy. (For instance, the MTU may be set to the Standard MTU if the link speed is below 1000 Mbps.)

MRU: Maximum Receive Unit. The size of the largest IP packet that can be received on an interface. This document doesn't use the term MRU, and assumes that the MRU is equal to the MTU.

MTU: Maximum Transfer Unit. The size of the largest IP packet that can be transmitted on an interface, considering hardware (and administrative) limitations.

Neighbor: Another node on a connected subnet. Neighbors are identified by the combination of a link address and an IP version.

The MTU may be set to different values for IPv4 and IPv6 administratively, but it is assumed that if a node has multiple IPv4 or IPv6 addresses, the MTU for each set of addresses is the same.

Neighbor MTU: The currently used MTU towards a neighboring node on a subnet. The Neighbor MTU reflects the current best understanding of the maximum packet size that can successfully be transmitted towards that neighbor.

Safe MTU: The maximum packet size that is assumed to work without testing. Defaults to the Standard MTU, but may be set to a subnet-wide higher or lower value administratively, or to a lower value using the MTU option in IPv6 Router Advertisements.

Standard MTU: The MTU specified in the relevant IPv4-over-... or IPv6-over-... document, which is 1500 for Ethernet ([RFC0894] and [RFC2464]).

4. Overview of operation

The mechanisms described in this document come into play when a node is connected to a subnet using an interface that supports an MTU size larger than the standard MTU size for that link type.

For each remote node connected to such a subnet, the local node maintains a neighbor MTU setting. The length of packets transmitted to a neighbor is always limited to the neighbor MTU size.

When a node starts communicating with another node on the same subnet, it follows the following procedure:

1. Initialization: the neighbor MTU is set to local maximum MTU for the interface used to reach the neighbor.

2. Discovery: learning the other node's MTU.

3. Probing: determining the maximum packet size that can successfully be transmitted to and and received from the other node, considering the (unknown) maximum packet size supported by the layer 2 infrastructure.

4. Monitoring: making sure that when large packets are transmitted, they are not silently discarded, for instance as the result of a layer 2 reconfiguration.

During the discovery and probing stages, the neighbor MTU is adjusted as new information becomes available. The monitoring stage is

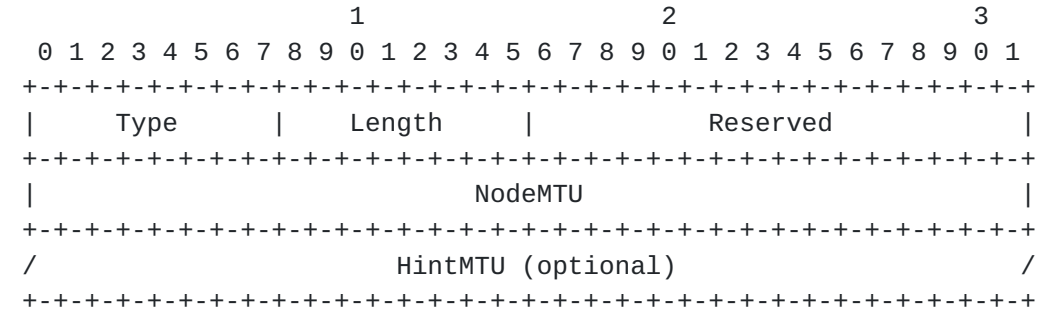
ongoing. If during the monitoring stage it is determined that large packets aren't successfully exchanged with the neighboring node, the neighbor MTU is set to the safe MTU and the node returns to the testing stage.

Unless administrative configuration or policy specifies otherwise, the link, IPv4 and IPv6 MTU sizes are set to the maximum supported by the hardware. This means that when TCP sessions are created, they carry a maximum segment size (MSS) option that reflects the larger-than-standard MTU.

5. The ND NODEMTU option

All MTU values are 32-bit unsigned integers in network byte order. All other values are also unsigned and in network byte order .

The MTU size and two flags are exchanged as an IPv6 Neighbor Discovery option. The new option, as well as the MTU value it avertises, are named "NODEMTU".



Type: TBD

Length: 1 or 2

Reserved: Set to 0 on transmission, ignored when received.

NodeMTU The maximum packet size the node wishes to receive on this interface.

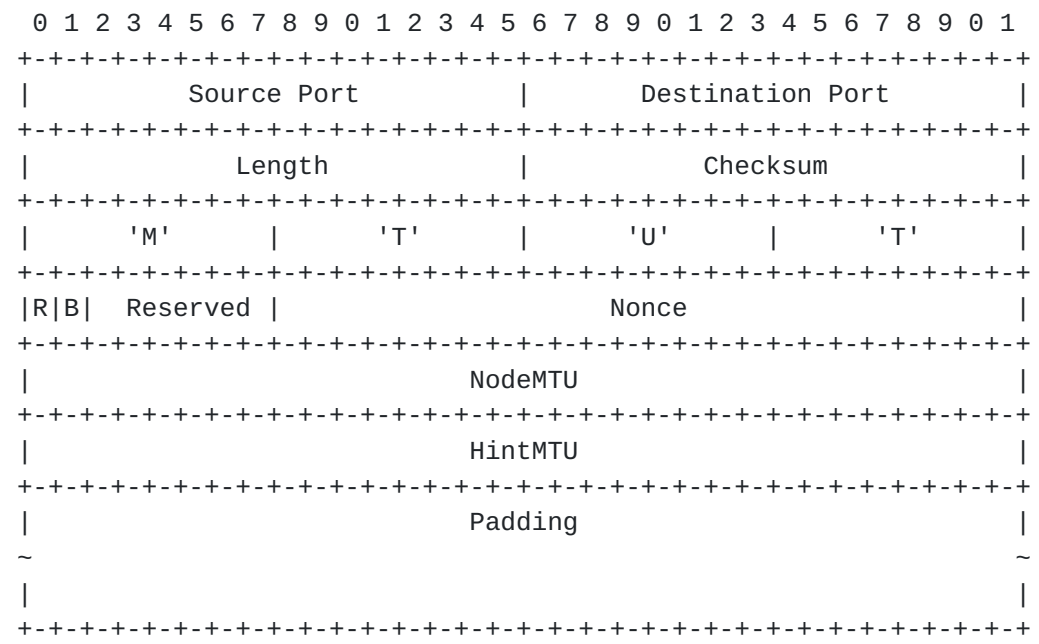
HintMTU The maximum packet size the node believes it can successfully receive on this interface at this time. If the HintMTU is equal to the NodeMTU or no value for HintMTU is known, this field may be omitted and the Length field is set to 1. If the HintMTU field is present, the Length field is set to 2.

When a node's interface speed changes, it MAY advertise a new MTU, but it SHOULD remain prepared to receive packets of the maximum size

advertised to neighbors previously (if the old maximum size is larger than the newly advertised one).

6. The MTUTEST packet format

The packets used to test whether large packets can be transmitted successfully and communicate status are sent using UDP ([RFC0768]). Their format is as follows:



Source port (UDP): For outgoing requests: an ephemeral port number. For replies: 1022. (16 bits.)

Destination port (UDP): For outgoing requests: 1022. For replies: the source port used in the request being replied to. (16 bits.)

Length (UDP): for IPv4 and IPv6 packets smaller than or equal to 65575 bytes, the length of the UDP segment. For IPv6 packets larger than 65575 bytes, 0 (as per [RFC2675]). (16 bits.)

Checksum (UDP): the UDP checksum. (16 bits.)

R: reply request flag. If set to 0, no reply is sent. If set to 1, the receiver is asked to send a reply. (1 bit.)

MTUT: The value corresponding to the ASCII string "MTUT", used to differentiate MTUTEST packets from other UDP packets that use port 1022. Packets with a value other than "MTUT" at the beginning of the UDP payload MUST be ignored. (32 bits.)

B: big reply request flag. If set to 0, replies are not padded. If set to 1, replies are padded to be the same size as the request. (1 bit.)

Reserved: set to 0 on transmission, ignored on reception. (6 bits.)

Nonce: a hard-to-guess value. (24 bits.)

NodeMTU: The maximum packet size that the sender is prepared to receive at this time. (32 bits.)

HintMTU: The maximum packet size that the sender believes it can successfully receive at this time. (32 bits.)

Padding: Filled with 0 or more all-zero bytes on transmission, ignored on reception.

In addition to the fields listed above, the following IP and link layer fields are taken into consideration:

Source link-layer address: On transmission: set automatically by the networking stack. On reception: used to identify a neighbor.

IP version: On transmission: set automatically by the networking stack. On reception: used to identify a neighbor. (The IP version may also be identified implicitly through the API without directly observing the version field.)

Time To Live / Hop Limit: On transmission: set to 255. On reception: if 255, the packet is processed. If other than 255, the packet is silently discarded. (To enforce that the protocol is only used within a local subnet.)

Source IP address: On transmission, for requests: set to the address the node intends to use to communicate with the neighbor. For replies: set to the destination IP address in the request being replied to. On reception: used to identify a neighbor.

Destination IP address: On transmission, for requests: set to the address the node intends to use to communicate with the neighbor. For replies: set to the source IP address in the request being replied to.

7. Changes to the RA MTU option semantics

Section 6.3.4 of [RFC4861] specifies:

"If the MTU option is present, hosts SHOULD copy the option's value into LinkMTU so long as the value is greater than or equal to the minimum link MTU and does not exceed the maximum LinkMTU value specified in the link-type-specific document"

This document changes the handling of the Router Advertisement MTU option such that it may also be used by routers to tell hosts that they SHOULD use an MTU larger than the LinkMTU and update their SafeMTU value. If multiple routers advertise different MTUs that are higher or lower than the standard MTU, behavior is undefined. MTU options containing the standard MTU SHOULD be ignored.

The ability to advertise a larger-than-standard MTU must be used with extreme care by nework administrators, as advertising an MTU size that exceeds the capabilities of routers or the layer 2 infrastructure will lead to reachability problems.

If the advertised larger-than-standard MTU is ignored or not supported by some hosts connected to the subnet, TCP will presumably still work because the MSS option ([RFC0793]) limits the size of transmitted TCP segments to what the receiver suports. However, non-TCP protocols that use large packets will likely fail. The most prominent example of this is DNS over UDP with EDNS0 when requesting large records, such as those used for DNSSEC ([RFC6891]).

8. The TCP MSS option

Hosts SHOULD advertise the maximum MTU size they are prepared to use on a link in the TCP MSS value, even during times when probing has failed: should larger neighbor MTUs be established later, it will not be possible to adjust the MSS for ongoing sessions.

9. Operation

9.1. Initialization

When an interface is activated, an appropriate local MTU is determined, based on hardware limitations and admnistrative settings. Additionally, a policy may be in place to constrain packet sizes when operating at lower bandwidths, to avoid excessive delays as queues of large packets build up and cause significant head-of-line blocking for subsequent time-sensitive packts. Also, layer 2 devices operating at lower interface speeds are less likely to support non-standard MTUs.

In the absense of operational experience, this document RECOMMENDS limiting the use of larger than standard MTUs to interfaces operating at 400 Mbps or faster; and if a larger MTU is used for interfaces

operating at lower speeds, a "mini jumbo" size of 1982 bytes or less is used for Ethernets.

For IPv4, the local MTU is limited to 65535 bytes. For IPv6, if [RFC2675] jumbograms are not supported, the local MTU is limited to 65575 bytes. These limits apply even if the interface hardware supports a larger MTU. IPv6 nodes that implement [RFC2675] jumbograms MAY use MTU sizes larger than 65575 bytes.

When the interface speed changes, the local MTU MAY be changed to reflect the new speed. However, the node SHOULD remain prepared to receive packets of the size of a previously advertised MTU.

The local MTU MAY be different for IPv4 and IPv6. The local MTU is the size used to calculate the value of the TCP MSS option. The HintMTU is set to undefined.

When sending Neighbor Solicitations and Neighbor Advertisements, a node includes its local MTU in the NodeMTU field of the NODEMTU option. If the size of the HintMTU is known, it is also included.

9.2. Probing

When a node starts communicating with a new IPv4 or IPv6 neighbor, the probing procedure is started. This can happen when ARP [RFC0826] or Neighbor Discovery messages are exchanged, or when an incoming TCP SYN is received.

The node sends a MTUTEST packet to the new neighbor and sets the neighbor MTU to the safe MTU. The MTUTEST packet has the local MTU in the NodeMTU field. If a hint MTU is known, it is included in the HintMTU field. The R and B flags are set to 0. No padding is included.

Upon reception of a Neighor Solicitation or a Neighbor Advertisement with the NODEMTU option or an MTUTEST packet, the node determines if the packet is received from a known neighbor IP address and a known neighor link layer address. If the values match the values stored for a known neighbor, no action occurs.

If the values match the values for a known link layer address and IP version, but an unknown IP address, the IP address is added to the list of IP addresses for the neighbor in question and the known neighbor MTU for the neighbor is applied to the new address.

If the NodeMTU matches the NodeMTU previously sent by a known neighbor but the HintMTU as a different non-zero value, the HintMTU is updated.

If the HintMTU sent by a known neighbor is 0, the neighbor MTU is set to the safe MTU, the HintMTU for the neighbor is set to unknown and the probing procedure is started.

If the combination of link layer address and IP version is unknown, the neighbor MTU is set to the safe MTU, the HintMTU is set to the HintMTU value in the packet and the probing procedure is started.

Before starting the probing procedure, a node compares its link layer address to the neighbor's link layer address. If the node's link layer address is numerically larger than the neighbor's link layer address, the node applies a waiting period before starting the probing procedure. The waiting period SHOULD be at least 250 milliseconds and at most 1 second.

The following is pseudo-code for a probing procedure. Note that it differens from the one outlined in [RFC4821]. The latter favors conservative probing because lost probes can't easily be differentiated from congestion losses, so lost probes are expensive. For this specification, successful probes waste bandwidth and losses are less problematic, so more aggressive probing and failing quickly is more appropriate.

```
Neighbor.ConfirmedTime = UNDEFINED

if LocalMTU > Neighbor.AdvertisedMTU
  let Max = Neighbor.AdvertisedMTU
else
  let Max = LocalMTU

# test with maximum supported packet size first
# and finish probing upon success
test (Max)
if Success:
   Neighbor.MTU = Max
   return

# maximum size doesn't work, now find
# what does work
# assumption: 256 works for IPv4, 1280 for IPv6
let WorksNo = Max
if IPv6:
  let Neighbor.ConfirmedMTU = 1280
if IPv4:
  let Neighbor.ConfirmedMTU = 256

# test with the hinted size
# if successful, this becomes the minimum for further tests
```

```
        # if unsuccessful, this becomes the maximum
        test (HintMTU)
        if Success:
          let Neighbor.ConfirmedMTU = HintMTU
        else
          let WorksNo = HintMTU

        # test the smallest usable size larger than
        # the standard MTU (if that size is still
        # in the range to be tested) so we avoid wasting
        # time probing non-jumbo-capable nodes
        if (StandardMTU + 8 > Neighbor.ConfirmedMTU and \
            StandardMTU + 8 < WorksNo)
          test (StandardMTU + 8)
          if Success:
            let Neighbor.ConfirmedMTU = StandardMTU + 8
          else
            let WorksNo = StandardMTU + 8

        # to establish an upper bound quickly,
        # test (320, 640, 1280, ) 2560, 5120, 10240, 20480, 40960, ...
        let Current = 320
        while (Current < WorksNo)
          if (Current > Neighbor.ConfirmedMTU)
            test (Current)
            if Success:
              let Neighbor.ConfirmedMTU = Current
            else
              let WorksNo = Current
          let Current = Current * 2

        # we have now established that
        # WorksNo =< Neighbor.ConfirmedMTU * 2

        # further testing is based on a list of hints.
        # there SHOULD be a mechanism for administrators
        # to add hints.
        #
        # hint sources:
        #   576: common PPP low delay
        #  1492: PPP over Ethernet [RFC2516]
        #  1500: Ethernet II
        #  1982: IEEE Std 802.3as-2006
        #  2304: IEEE 802.11
        #  2482: Fibre Channel over Ethernet (FCoE)
        # [CATALYST]:
        # 9216, 8092, 1600, 1998, 2000, 1546, 1530, 17976, 2018
        # sizes observed by the author:
```

```
        # 576, 1982, 4070, 9000, 16384, 64000
        let Hints = 576, 1492, 1530, 1982, 2304, 4070, 8092, 9000, \
                    16384, 32000, 64000

        foreach Size in Hints
          if Size > Neighbor.ConfirmedMTU and Size < WorksNo
            test (Size)
            if Success:
              let Neighbor.ConfirmedMTU = Size
            else
              let WorksNo = Size

        # finished testing, maximum working packet size
        # is now known to within about a factor 1.5,
        # depending on the number of hints

        if Neighbor.ConfirmedTime <> UNDEFINED
          # we got at least one probe back, use discovered MTU
          Neighbor.MTU = Neighbor.ConfirmedMTU
        else
          # we never got any probes back, neighbor probably does
          # not implement MTUTEST protocol, so we use the safe MTU
          Neighbor.MTU = SafeMTU

        # done!
        return

        # sending probes
        function test (Size)

        # wait 20 milliseconds between sending probes
        let MsecSinceProbe = now () - ProbeTime

        if (MsecSinceProbe < 20)
          sleep (20 - MsecSinceProbe)

        # create probe, request reply (but not a big one)
        let Probe.TTL = 255
        let Probe.ReplyFlag = 1
        Let Probe.BigFlag = 0
        Let Nonce = rand ()
        Let Probe.Nonce = Nonce
        let Probe.NodeMTU = LocalMTU
        let Probe.HintMTU = HintMTU
        let Probe.Padding = pad (Size - sizeof (Probe))
        send (Probe)

        let ProbeTime = now ()
```

```
# wait 2000 milliseconds for reply
# (this also avoids sending packets that are too large more
# than once every two seconds)
let Success = receive (Reply, 2000)

if not Success
  return false

if not (Reply.TTL = 255 and Reply.Nonce = Nonce
  and Reply.LinkAddress = Neighbor.LinkAddress)
    return false

# valid reply received
# note that Neighbor.MTU is not updated yet,
# this happens after probing has finished
Neighbor.ConfirmedMTU = Reply.NodeMTU
Neighbor.ConfirmedTime = now ()
Neighbor.HintMTU = Reply.HintMTU;
if HintMTU < Size
  HintMTU = Size
return true
```

If at any time an unsolicited packet arrives from the neighbor and the ConfirmedMTU of that neighbor is smaller than the size of the packet received, the HintMTU for the neighbor is set to the size of the received packet and a probe of that size may be sent. However, as the maximum size of incoming packets may be different than the maximum supported size of outgoing packets, reception of a large packet is not sufficient to update the ConfirmedMTU. The packets that update the HintMTU do not have to be MTUTEST protocol packets.

There are no retransmissions. Both nodes run the probing procedure, so there are two opportunities to succeed. However, if both fail to determine the maximum packet size that can be used because of lost packets, the hosts will have to use a smaller packet size.

It is assumed that the maximum packet size that A can send to B is the same as the maximum packet size that B can send to A. As such, the reception of a large packet is treated the same as receiving an acknowledgment for a sent large packet.

9.3. Monitoring

Once a working neighbor MTU is found, large packets can be exchanged. Presumably, this situation will persist indefinitely. However, it is possible that the network is reconfigured and then no longer supports the MTU used between two nodes. The aim of the monitoring phase is

to detect this when it happens and establish a working MTU value before sessions time out.

For each neighbor (as defined by a unique combination of link layer address and IP version) with a neighbor MTU larger than the safe MTU, the ability to successfully send or receive large packets is monitored. In the monitoring phase, a node tracks whether it sends any packets larger than the safe MTU to a neighbor and whether it receives either acknowledgments for those packets, or it receives packets of length neighbor MTU from that neighbor. (So acknowledged outgoing packets don't have to be the maximum size supported to/from the neighbor, but incoming packets do.)

The ability to track acknowledgment of non-MTUTEST packets is not required. However, it is expected that hosts will be able to do this for TCP packets because the TCP state is readily available.

Monitoring is happens in intervals. This document RECOMMENDS that this interval is between 25 and 35 seconds for hosts and between 35 and 45 seconds for routers. At the end of each monitoring interval, if acknowledgments or large packets were received, everything is fine and the neighbor confirmed time is updated.

At the end of a monitoring interval, if no large packets were sent, everything is fine and nothing happens.

At the end of a monitoring interval, if large packets were sent, but no acknowledgments or incoming maximum size packets were seen, there may have been a network reconfiguration that has made it impossible for large packets to be transmitted successfully between the two nodes. To determine whether this is the case, the node sends an MTUTEST packet with lenght neighbor MTU. The R flag is set to 1 and the B flag SHOULD be set to 0. A random nonce and the local MTU and the hint MTU are included.

The node waits 2 seconds for a reply. If there is no reply, the probe is retransmitted and the node waits 4 seconds for a reply. If after 4 seconds there is still no reply, the node sets the hint MTU to 0 and reinitializes all of the neighbor's MTU-related information to initial values. Most notably, this means that the neighbor MTU is set to the safe MTU.

If the node sets is own hint MTU to 0 or receives a hint MTU of 0 from a neighbor using an ND or MTUTEST packet, the node MAY start sending probes to other neighbors before the monitoring interval expires. However, nodes SHOULD limit the number of probes for all neighbors combined to no more than one every two seconds. If a node has many neighbors and sending probes at one every two seconds would

take too long, it MAY reset the neighbor MTUs of all of its neighbors to the safe MTU without sending probes if at least two neighbors appear to be affected by a reduction of the maximum working packet size.

9.4. Neighbor MTU garbage collection

The MTU size for a neighbor is garbage collected along with a neighbor's link address in accordance with regular ARP and neighbor discovery timeouts. Additionally, a neighbor's MTU size is reset to unknown after dead neighbor detection declares a neighbor "dead".

10. Applicability

As discussed in annex B, all larger packets, but especially very large packet sizes have the potential to be problematic in various ways. However, jumboframes of 9000 or 9216 bytes have been supported by various vendors for a long time. As such, larger MTUs of 9 kilobytes seem safe enough for larger scale experimentation at this time, but experiments with packet sizes larger than 11 kilobytes are best done in confined and closely monitored situations.

11. IANA considerations

IANA is requested to assign a neighbor discovery option type value.

[TO BE REMOVED: This registration should take place at the following location: http://www.iana.org/assignments/icmpv6-parameters

UDP port 1022 is used in accordance with [RFC4727]. Presumably, unlike an ND option type value, a UDP port would be relatively easy to change when experimentation makes way for production deployment.

12. Security considerations

Generating false neighbor discovery and MTUTEST packets with large MTUs may lead to a denial-of-serve condition, just like the advertisement of other false link parameters. Requests are large and replies typically short to avoid the MTUTEST protocol being used as an amplification vector. The nonce is used together with the ephemeral UDP port number to make sure that malicious nodes cannot generate a reply to a request in the blind. Enforcement of the value 255 for Hop Limit makes sure that off-link attackers can't use the protocol to influence packet sizes remotely.

A malicious node may negotiate the use of large packets and cause head-of-line blocking, especially on slower links. However, this can

only happen if the neighbor is prepared to use large packets in the first place.

13. Acknowledgements

This document benefited from feedback by Dave Thaler, Jari Arkko, Joe Touch, Pat Thaler, David Black, Brian Carpeter, Fred Templin, Jeffrey Hammond, Mikael Abrahamsson and others.

14. References

14.1. Normative References

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <http://www.rfc-editor.org/info/rfc768>.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <http://www.rfc-editor.org/info/rfc793>.

[RFC0826] Plummer, D., "Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware", STD 37, RFC 826, DOI 10.17487/RFC0826, November 1982, <http://www.rfc-editor.org/info/rfc826>.

[RFC0894] Hornig, C., "A Standard for the Transmission of IP Datagrams over Ethernet Networks", STD 41, RFC 894, DOI 10.17487/RFC0894, April 1984, <http://www.rfc-editor.org/info/rfc894>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2464] Crawford, M., "Transmission of IPv6 Packets over Ethernet Networks", RFC 2464, DOI 10.17487/RFC2464, December 1998, <http://www.rfc-editor.org/info/rfc2464>.

[RFC2675] Borman, D., Deering, S., and R. Hinden, "IPv6 Jumbograms", RFC 2675, DOI 10.17487/RFC2675, August 1999, <http://www.rfc-editor.org/info/rfc2675>.

[RFC2992] Hopps, C., "Analysis of an Equal-Cost Multi-Path Algorithm", RFC 2992, DOI 10.17487/RFC2992, November 2000, <http://www.rfc-editor.org/info/rfc2992>.

[RFC4727] Fenner, B., "Experimental Values In IPv4, IPv6, ICMPv4, ICMPv6, UDP, and TCP Headers", RFC 4727, DOI 10.17487/RFC4727, November 2006, <http://www.rfc-editor.org/info/rfc4727>.

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, DOI 10.17487/RFC4821, March 2007, <http://www.rfc-editor.org/info/rfc4821>.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <http://www.rfc-editor.org/info/rfc4861>.

[RFC6891] Damas, J., Graff, M., and P. Vixie, "Extension Mechanisms for DNS (EDNS(0))", STD 75, RFC 6891, DOI 10.17487/RFC6891, April 2013, <http://www.rfc-editor.org/info/rfc6891>.

[ETHERNETII] Digital Equipment Corporation, Intel Corporation, Xerox Corporation, ""The Ethernet - A Local Area Network", September 1980, <http://research.microsoft.com/en-us/um/people/gbell/Digital/Ethernet%20Blue%20Book.pdf>.

14.2. Informative References

[RFC2516] Mamakos, L., Lidl, K., Evarts, J., Carrel, D., Simone, D., and R. Wheeler, "A Method for Transmitting PPP Over Ethernet (PPPoE)", RFC 2516, DOI 10.17487/RFC2516, February 1999, <http://www.rfc-editor.org/info/rfc2516>.

[IEEE.802.3AS_2006] IEEE, "IEEE Standard for Information Technology Telecommunications and Information Exchange Between Systems Local and Metropolitan Area Networks Specific Requirements Part 3: Carrier Sense Multiple Access With Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications Amendment 3: Frame Format Extensions", IEEE 802.3as-2006, DOI 10.1109/ieeestd.2006.248146, November 2006, <http://ieeexplore.ieee.org/servlet/opac?punumber=4014413>.

[IEEE.802.3_2012]
IEEE, "802.3-2012", IEEE 802.3-2012,
DOI 10.1109/ieeestd.2012.6419735, January 2013,
<http://ieeexplore.ieee.org/servlet/opac?punumber=6419733>.

[CRC] Jain, R., "Error Characteristics of Fiber Distributed Data Interface (FDDI), IEEE Transactions on Communications", August 1990.

[CATALYST]
Cisco, "Jumbo/Giant Frame Support on Catalyst Switches Configuration Example",
<http://www.cisco.com/c/en/us/support/docs/switches/catalyst-6000-series-switches/24048-148.html>.

Appendix A. Document and discussion information

The latest version of this document will always be available at http://www.muada.com/drafts/. Please direct questions and comments to the int-area mailinglist or directly to the author.

Appendix B. Advantages and disadvantages of larger packets

Although often desirable, the use of larger packets isn't universally advantageous for the following reasons:

1. Clock skew

2. ECMP over paths with different MTUs

3. Increased delay and jitter

4. Increased reliance on path MTU discovery

5. Increased packet loss through bit errors

6. Increased risk of undetected bit errors

B.1. Clock skew

Ethernet hardware has to compensate between clocking differences between the sender and receiver though a FIFO buffer. As packets get larger, more buffer capacity is required. This places a limit on packet sizes.

As jumboframes have been widely supported sinze the introduction of Gigabit Ethernet, an in the absense of information to the contrary,

it seems safe to assume that the packet sizes that may be set administratively fall within the capabilities of the hardware. Administrators are encouraged to monitor the fraction of packets lost from different types of corruption and adjust MTU sizes accordingly.

B.2. ECMP over paths with different MTUs

Should Equal Cost Multipath [RFC2992] be in effect between two nodes implementing this specification, with the different paths having different MTUs, then there is a high risk that probing will detect the larger of the supported MTU sizes but some data packets will flow over the path with the smaller MTU size. In this situation, packets will be lost consistently and the protocol will not be able to recover.

As such, configuring paths used for ECMP with different MTU sizes MUST be avoided.

B.3. Delay and jitter

An low-bandwidth links, the additional time it takes to transmit larger packets may lead to unacceptable delays. For instance, transmitting a 9000-byte packet takes 7.23 milliseconds at 10 Mbps, while transmitting a 1500-byte packet takes only 1.23 ms. Once transmission of a packet has started, additional traffic must wait for the transmission to finish, so a larger maximum packet size immediately leads to a higher worst-case head-of-line blocking delay, and thus, to a bigger difference between the best and worst cases (jitter). The increase in average delay depends on the number of packets that are buffered, the average packet size and the queuing strategy in use. Buffer sizes vary greatly between implementations, from only a few buffers in some switches and on low-speed interfaces in routers, to hundreds of megabytes of buffer space on 10 Gbps interfaces in some routers.

If we assume that the delays involved with 1500-byte packets on 100 Mbps Ethernet are acceptable for most, if not all, applications, then the conclusion must be that 15000-byte packets on 1 Gbps Ethernet should also be acceptable, as the delay is the same. At 10 Gbps Ethernet, much larger packet sizes could be accommodated without adverse impact on delay-sensitive applications. At below 100 Mbps, larger packet sizes are probably not advisable.

When very tight QoS bounds are required, it may be appropriate to limit MTU sizes and forego larger MTUs. With IPv6 this can be accomplised by advertising a limited MTU size in Router Advertisements. With IPv4, it is necessary to configure each node to limit its MTU size.

B.4. Path MTU Discovery problems

PMTUD issues arise when routers can't fragment packets in transit because the DF bit is set or because the packet is IPv6, but the packet is too large to be forwarded over the next link, and the resulting "packet too big" ICMP messages from the router don't make it back to the sending host. If there is a PMTUD black hole, this will typically happen when there is an MTU bottleneck somewhere in the middle of the path. If the MTU bottleneck is located at either end, the TCP MSS (maximum segment size) option makes sure that TCP packets conform to the smallest MTU in the path. PMTUD problems are of course possible with non-TCP protocols, but this is rare in practice because non-TCP protocols are generally not capable of adjusting their packet size on the fly and therefore use more conservative packet sizes which won't trigger PMTUD issues.

Taking the delay and jitter issues to heart, maximum packet sizes should be larger for faster links and smaller for slower links. This means that in the majority of cases, the MTU bottleneck will tend to be at, or close to, one of the ends of a path, rather than somewhere in the middle, as in today's internet, the core of the network is quite fast, while users usually connect to the core at lower speeds.

A crucial difference between PMTUD problems that result from MTUs smaller than the de facto standard 1500 bytes and PMTUD problems that result from MTUs larger than 1500 bytes is that in the latter case, only the party that's actually using the non-standard MTU is affected. This puts potential problems, the potential benefits and the ability to solve any resulting problems in the same place: it's always possible to revert to a 1500-byte MTU if PMTUD problems can't be resolved otherwise.

Considering the above and the work that's going on in the IETF to resolve PMTUD issues as they exist today, increasing MTUs where desired doesn't seem to involve undue risks.

B.5. Packet loss through bit errors

All transmission media are subject to bit errors. In many cases, a bit error leads to a CRC failure, after which the packet is lost. In other cases, packets are retransmitted a number of times, but if error conditions are severe, packets may still be lost because an error occurred at every try. Using larger packets means that the chance of a packet being lost due to errors increases. And when a packet is lost, more data has to be retransmitted.

Both per-packet overhead and loss through errors reduce the amount of usable data transferred. The optimum tradeoff is reached when both

types of loss are equal. If we make the simplifying assumption that the relationship between the bit error rate of a medium and the resulting number of lost packets is linear with packet size for reasonable bit error rates, the optimum packet size is computed as follows:

packet size = sqrt(overhead bytes / bit error rate)

According to this, the optimum packet size is one or more orders of magnitude larger than what's commonly used today. For instance, the maximum BER for 1000BASE-T is 10^-10, which implies an optimum packet size of 312250 bytes with Ethernet framing and IP overhead.

B.6. Undetected bit errors

Nearly all link layers employ some kind of checksum to detect bit errors so that packets with errors can be discarded. In the case of Ethernet, this is a frame check sequence in the form of a 32-bit CRC. Assuming a strong frame check sequence algorithm, a 32-bit checksum suggests that there is a 1 in 2^32 chance that a packet with one or more bit errors in it has the same checksum as the original packet, so the bit errors go undetected and data is corrupted. However, according to [CRC] the CRC-32 that's used for FDDI and Ethernet has the property that packets between 375 and 11453 bytes long (including) have a Hamming distance of 4. (Smaller packets have a larger Hamming distance, larger packets a smaller Hamming distance.) As a result, all errors where only a single bit is flipped, two bits are flipped or three bits are flipped, will be detected, because they can't result in the same CRC as the original packet. The probability of a packet having undetected bit errors can be approximated as follows for a 32-bit CRC:

PER = (PL * BER) ^ H / 2^32

Where PER is the packet error rate, BER is the bit error rate, PL is the packet length in bits and H is the Hamming distance. Another consideration is the impact of packet length on a multi-packet transmission of a given size. This would be:

TER = transmission length / PL * PER

So

TER = transmission length / (PL ^ (H - 1) * BER ^ H) / 2^32

Where TER is the transmission error rate.

In the case of the Ethernet FCS and a Hamming distance of 4 for a large range of packet sizes, this means that the risk of undetected errors goes up with the cube of the packet length, but goes down with the fourth power of the bit error rate. This suggest that for a given acceptable risk of undetected errors, a maximum packet size can be calculated from the expected bit error rate. It also suggests that given the low BER rates mandated for Gigabit Ethernet, packet sizes of up to 11453 bytes should be acceptable.

Additionally, unlike properties such as the packet length, the frame check sequence can be made dependent on the physical media, so in the future it should be possible to define a stronger FCS in future Ethernet standards, or to negotiate a stronger FCS between two stations on a point-to-point Ethernet link (i.e., a host and a switch or a router and a switch).

B.7. Interaction TCP congestion control

TCP performance is based on the inverse of the square of the packet loss probability. Using larger and thus fewer packets is therefore a competitative advantage. Larger packets increase burstiness, which can be problematic in some circumstances. Larger packets also allow TCP to ramp up its transmission speed faster, which is helpful on fast links, where large packets will be more common. In general, it would seem advantageous for an individual user to use larger packets, but under some circumstances, users using smaller packets may be put at a slight disadvantage.

B.8. IEEE 802.3 compatibility

According to the IEEE 802.3 standard ([IEEE.802.3_2012]), the field following the Ethernet addresses is a length field. However, [RFC0894] uses this field as a type field. Ambiguity is largely avoided by numbering type codes above 2048. The mechanisms described in this memo only apply to the standard [RFC0894] and [RFC2464] encapsulation of IPv4 and IPv6 in Ethernet, not to possible encapsulations of IPv4 or IPv6 in IEEE 802.3/IEEE 802.2 frames, so there is no change to the current use of the Ethernet length/type field.

The 2006 revision of IEEE 802.3 ([IEEE.802.3AS_2006]) adds "frame expansion" to 2000 bytes (allowing for 1982-byte IP packets). As a result, layer 2 networks supporting MTUs of 1982 bytes are becoming more common. However, as [RFC0894] and [RFC2464] (encapsulation of IPv4 and IPv6 in Ethernet) are based on [ETHERNETII]), the IEEE 802.3 standard has little bearing on the problem at hand.

B.9. Conclusion

Larger packets aren't universally desirable. The factors that factor into the decision to use larger packets include:

- A link's bit error rate
- The number of bits per symbol on a link and hence the likelihood of multiple bit errors in a single packet
- The strength of the frame check sequence
- The link speed
- The number of buffers
- Queuing strategy
- Number of sessions on shared links and paths

This means that choosing a good maximum packet size is, initially at least, the responsibility of hardware builders. A conservative approach may be called for, but even under conservative assumptions, 9000-byte jumboframes on Gigabit Ethernet links seem reasonable.

Author's Address

Iljitsch van Beijnum
Institute IMDEA Networks
Avda. del Mar Mediterraneo, 22
Leganes, Madrid 28918
Spain

Email: iljitsch@muada.com