Network Working Group
Internet-Draft
Intended status: Experimental
Expires: January 13, 2011

I. van Beijnum
IMDEA Networks
July 12, 2010

Extensions for Multi-MTU Subnets
draft-van-beijnum-multi-mtu-03

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 January 13, 2011.

Table of Contents

1. Introduction

Some protocols inherently generate small packets. Examples are VoIP, where it's 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 rarely 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 on 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 switches connected to a subnet. Also, the MTU size for a subnet is limited to that of the least capable router, host or switch.

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 several new options and messages for both IPv6 and IPv4:

1. A neighbor discovery option that allows nodes to inform their neighbors of the maximum packet sizes they are prepared to receive

2. An extension to the ARP packet format that allows nodes to inform their neighbors of the maximum packet sizes they are prepared to receive

3. A probe/verification message that allows nodes to determine whether jumboframes can be received successfully by the next hop

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.

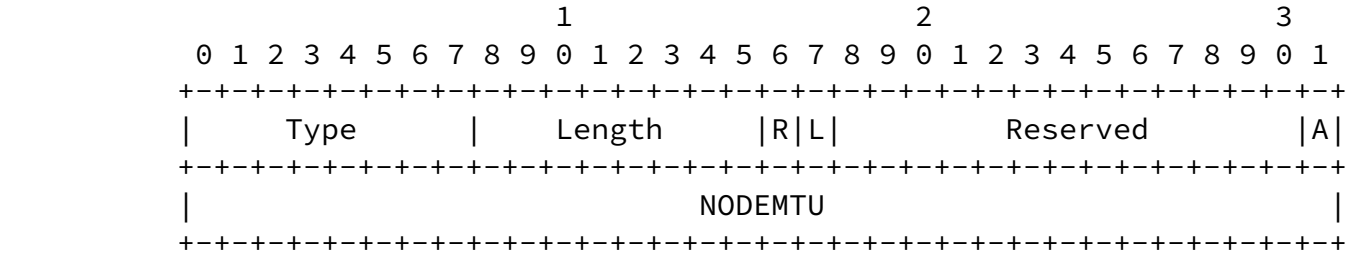
3. Protocol messages and options

3.1. The ND/ARP 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. Troughout this document, the term "MTU" is used to denote the maximum packet size that can be sent or received. The term "MRU" (maximum receive unit) is not used. The "standard MTU" or "standard maximum size" refers to the MTU size specified in the IP-over-... or IPv6-over-... document for the link used, which would be 1500 for ethernet.

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". For IPv4 operation, the NODEMTU option is appended to ARP messages, with optional padding between the ARP message and the MTU option. Upon reception of ARP messages, the

receiving node checks whether the ARP message is 8 or more bytes longer than a standard ARP message. If so, the NODEMTU option is ignored if the Type and Length fields contain values other than the ones listed below, or if the MTU is smaller than the standard value for the link type.



Type: TBD

Length: 1

R (router): Set to 1 if the node is a router, set to 0 if the node is a host or routing functionality is currently disabled.

L (large packet detect): Set to 1 if the node is capable of

determining the largest size of packets recently received from a link address, set to 0 if the node requires explicit probe messages.

Reserved: Set to 0 on transmission, MUST be ignored on reception.

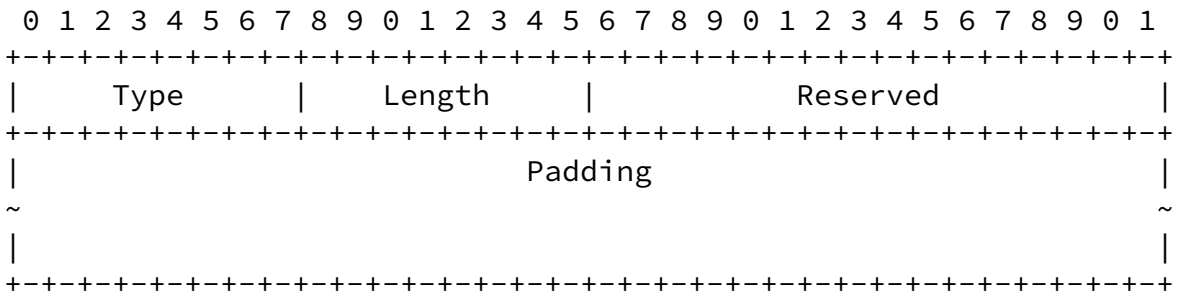
A (acknowledgment): Set to 1 if the node received a packet larger than the interface MTU from the node this packet is addressed to in the last 10 seconds.

NODEMTU The maximum packet size the node wishes to receive on this interface at this time.

When a node's interface speed changes, it MAY advertise adjusted per-neighbor MTUs, but it SHOULD remain prepared to receive packets of the maximum size indicated to neighbors previously (if this maximum size is larger than the newly adjusted one).

3.2. The IPv6 ND padding option

The format of the neighbor discovery padding option is as follows:



Type: TBD

Length: see below.

Reserved: set to 0 on transmission, ignored on reception.

Padding: 0 or more all-zero bytes.

There are two possible ways to determine the value of the length field in the padding option:

1. Set it to 0. Since the option is in fact larger than 0, this means that nodes that don't implement the option will silently discard the packet. Setting the length to 0 makes it possible to have packets with the padding option that aren't a multiple of 8 bytes long. Since there is now no way to determine where the next option begins, if the length is set to 0, the padding option MUST be the last option.

2. If the intended packet length allows a valid value for the length field, the length field MAY be set to that value. The node MAY reduce the size of the intended packet to accommodate the requirement that the size field is a multiple of 8 bytes. I.e., if the intended packet size is 4470 bytes with 40 and 24 bytes for the IPv6 and neighbor solicitation headers, respectively, the padding option would have to be 4406 bytes long, which can't be expressed in the length field. The node may choose to use a packet size of 4464 instead, which results in a length field value of 550. This of course means that subsequent data packets MUST be no larger than 4464 bytes.

Nodes that support probing MUST support reception of both types of probes, but MAY be limited to generating only one type.

Since presumably, some equipment may react badly to a large number of out-of-spec packets, it's important that nodes limit the number of oversized packets to destinations that aren't yet known to be capable of receiving them. An upper limit would be to allow only 5 unacknowledged oversized packets per 300 second period.

3.3. IPv4 ethernet jumbo ARP message

Due to lack of neighbor discovery, with IPv4, it's necessary to use ARP to probe for non-standard MTU capabilities. This is done by simply probing with an ARP packet padded to the desired size. If a reply comes back, the neighbor supports the probed MTU size. A NODEMTU option MAY or MAY NOT be present in the last 8 bytes of the jumbo ARP message. Nodes MUST take care to include either a valid

NODEMTU option or bytes that can't be mistaken for a NODEMTU option.

3.4. Changes to the RA MTU option semantics

There may be an MTU option in IPv6 router advertisements. When this option is present, hosts MUST use the value in this option only as a replacement for the standard link MTU size. So for multicast packets and packets sent to nodes for which there is no known NODEMTU, the value in the MTU option is used as the maximum packet size. But if a NODEMTU is known for a node on the link, the NODEMTU is used, NOT the value in the RA MTU option.

4. Operation

Basic operation is as follows: nodes advertise their interface MTU in a neighbor discovery option or in ARP messages. So for communication between two nodes implementing this specification, each knows what the maximum packet size is that the other node supports, so the minimum of the local and the remote MTU is used when sending packets.

Unfortunately, there is the complication that layer 2 devices (switches/bridges) may have a smaller maximum packet size, so packets larger than the standard maximum size may be lost. In order to avoid this issue, this memo specifies a "trust, but verify" approach: whenever packets larger than the standard size are supported between two nodes, each periodically verifies that the other is still capable of receiving packets of the negotiated size. It does this by sending a probe message of the maximum negotiated size, followed by If the verification fails, the node sends a new neighbor advertisement with a reduced MTU size.

A number of optimizations reduce the amount of signaling traffic where possible. Most of the optimizations are optional.

In the case of a host implementing packetization layer path MTU discovery [RFC4821] for all transport protocols that can generate packets larger than the standard size, the use of outgoing probe and verification messages is unnecessary. However, such a host MUST still process incoming probe and verification messages.

The first optimization is for senders that have the capability to

determine whether they sent packets that are larger than the standard size, to only send a probe message and a verification message when a data packet larger than the standard size was sent recently.

A further optimization may be applied when both the sender and the receiver have the capability to determine whether they sent/received data packets that are larger than the standard size. If both the sender and the receiver have this capability, as indicated by flags in the neighbor discovery option, no explicit MTU probe messages are sent, just a verification message.

A final optimization applies between a host and a router. In that case, the host may assume the responsibility for probing with large packets in both directions. This reduces the control channel processing on the router, and allows for the possibility to forego probing when there are no active transport sessions that are capable of generating larger than standard packets.

4.1. Managing neighbor MTUs

The following does not apply to hosts that support [RFC4821] or a similar mechanism for all transport protocols that can send larger than standard packets. For instance, if a host implements [RFC4821] for TCP and limits UDP packets and packets using other transport protocols to 1500 bytes on its ethernet interface, the host is not requred to perform any probing and per-destination path MTUs can be maintained at the TCP level. It must still respond to incoming probes. Routers are never exempt from what follows.

Along with neighbor's link addresses, a node caches an MTU value for each neighbor. This value starts out being undefined. Whenever a packet must be sent (this includes packets that are forwarded), the node consults the neighbor MTU cache. If the cached value is undefined, it applies the interface MTU that is in effect on MTU-related actions such as fragmentation or the generation of "too big" messages.

Whenever a value is entered into the neighbor MTU cache, this value is marked as "tentative" and the node MUST start a clock that times out after 500 milliseconds. After 500 milliseconds (or less, depending on the implementation), if the value in the MTU cache is still tentative, it reverts back to being undefined. Values enter the neighbor cache after receiving the NODEMTU option in ND or ARP messages. In this case, the cache is initalized with the minimum of the local and remote NODEMTU values and the clock is started. If no such option is present in ND or ARP messages, the node may insert the standard MTU value + 1 rounded up to the nearest multiple of 8. In

this case, the clock is also started. If timer resources are not available, the neighbor MTU value remains undefined.

At this point, the node sends a probe message that is the size of the value in the neighbor MTU cache. If this probe message is answered, the neighbor's MTU value in the neighbor MTU cache is marked as "valid" and the timer is stopped.

If the probe wasn't answered, or probing started from just above the standard MTU, after some time (such as 30 seconds) a new probe MAY be sent. For unanswered probes, new probes are larger, for answered probes the new probe is larger. If the new probe is answered, the size of the probe is entered in the neighbor MTU cache as a "valid" value. Probing MAY continue for several iterations, but implementers are encouraged to limit probing rather than exhaustively search for the exact supported neighbor MTU value.

4.2. Host-to-host keepalives

When hosts have active communication sessions with other hosts on the same subnet, they send periodic probes to determine whether large packets continue to be received. Hosts MUST NOT send probes when there are no active communication sessions and SHOULD NOT send probes when there are no active communication sessions that support larger than standard packet sizes. For instance, if the host only supports larger-than-standard packet sizes over TCP, and there are no TCP sessions where the remote host indicated that it supports larger-than-standard packet sizes through the MSS option, probing SHOULD NOT be performed.

The probe interval is randomized between 8 and 10 seconds. A host SHOULD NOT send probes if it has not sent any packets larger than the interface MTU size during the previous probe interval. Probes are ARP or neighbor solicitation messages padded to the cached neighbor MTU size. A timer is initialized to 21 seconds (or a slightly larger value, with a maximum of 35 seconds) when a probe is sent. The timer is NOT reinitialized when new probes are sent. The timer is stopped when a probe is answered by an ARP reply or neighbor advertisement. This message does not have to be padded. If the timer is not stopped by an incoming probe reply and it expires, the neighbor MTU cache is cleared and becomes undefined. The next probe is sent no earlier than 8 seconds after the last ARP or neighbor advertisement from the neighbor has been received.

The following is optional:

If the neighbor included the L flag set to 1 in its NODEMTU option, the host MAY send probes as regular ARP or neighbor solicitation

packets, without padding. In this case, ARP replies or neighbor advertisements are only considered valid probe replies when they have a NODEMTU option with the A flag set to 1.

If the host detected that it received a packet larger than the interface MTU in the last 7 seconds, it MAY send an unsolicited probe reply, which consists of an ARP reply or a neighbor advertisement.

Replies to probes SHOULD and unsolicited probe replies MUST have a NODEMTU option with the A bit set.

4.3. Router-to-router keepalives

The probe interval is randomized between 8 and 10 seconds. A router SHOULD NOT send probes if it has not sent any packets larger than the interface MTU size during the previous probe interval. Probes are ARP or neighbor solicitation messages padded to the cached neighbor MTU size. A timer is initialized to 21 seconds (or a slightly larger value, with a maximum of 35 seconds) when a probe is sent. The timer is NOT reinitialized when new probes are sent. The timer is stopped when a probe is answered by an ARP reply or neighbor advertisement. This message MUST NOT be padded. If the timer is not stopped by an incoming probe reply and it expires, the neighbor MTU cache is cleared and becomes undefined.

4.4. Host-to-router keepalives

The probe interval is randomized between 8 and 10 seconds. Probes are ARP or neighbor solicitation messages padded to the cached neighbor MTU size. The destination IP address is the host's own IP address, the link address is the router's link address. A timer is initialized to 21 seconds (or a slightly larger value, with a maximum of 35 seconds) when a probe is sent. The timer is NOT reinitialized when new probes are sent. The timer is stopped when a probe is answered by an ARP reply or neighbor advertisement. This message MUST be the padded message originally sent by the host itself. If the timer is not stopped by an incoming probe reply and it expires, the neighbor MTU cache is cleared and becomes undefined. Also, a neighbor advertisement or ARP reply is sent with a NODEMTU option

that contains the current interface MTU. After some time, such as 15 minutes, the host MAY attempt probing for larger than standard MTU sizes again.

4.5. Router-to-host keepalives

Routers do not send keepalives to hosts. Routers MUST adjust their cached neighbor MTU value based on the NODEMTU option in unsolicited neighbor advertisements or ARP replies.

4.6. Determining the MTU

Nodes SHOULD NOT blindly advertise the maximum MTU that their hardware is capable of. On slow links, a large MTU can easily reduce performance. In general, hosts SHOULD limit the MTU they advertise and impose on packets they send to the standard MTU size on links operating at speeds of 50 Mbps or slower. On links operating at speeds of 500 Mbps and higher, MTUs of 9000 bytes or even as large as 64 kilobytes presumably won't cause problems. Between 50 and 500 Mbps, larger than standard MTUs SHOULD be used with care. For instance, by limiting MTUs to 9000 bytes and only on full duplex links with low bit error rates (which would exclude wireless links).

4.7. Probe considerations

In cases where the neighbor's MTU was advertised in a NODEMTU option, it makes sense to try with this size or the local MTU, whichever is smaller. If that probe fails or the neighbor's MTU is unknown, the best choice for a probe size would be the smallest possible non-standard MTU. This could be the StandardMTU + 1, or a slightly larger value that represents the first larger size that is actually useful, such as 1508 or 1520 for ethernet. Failure at this size wastes relatively little bandwidth and indicates that further probes are unnecessary. If this probe is successful, further choices for the probe size may be common MTU sizes such as 1508, 1530, 1536, 1546, 1998, 2000, 2018, 4464, 4470, 8092, 8192, 9000, 9176, 9180, 9216, 16384, 17976, 64000 and 65280 bytes. (These values were observed in vendor documentation and hands-on experience.)

A further consideration is that there is little value in sending many probes to discover a few extra bytes of MTU, and using multiples of 8 bytes may streamline copying of data and makes IPv6 probing easier.

So values to test if the NODEMTU fails but 1504 succeeds could be 1992, 4464, 8088, 9000, 16384 and 64000.

Probes MUST be sent as unicast.

4.8. 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".

5. 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.

6. IANA considerations

IANA is requested to assign two neighbor discovery option type values.

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

7. Security considerations

Generating false neighbor discovery and ARP packets with large MTUs may lead to a denial-of-serve condition, just like the advertisement of other false link parameters.

8. Acknowledgements

This document benefited from feedback by Dave Thaler, Jari Arkko, Joe Touch and others.

9. References

9.1. Normative References

[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, November 1982.

[RFC0894] Hornig, C., "Standard for the transmission of IP datagrams over Ethernet networks", STD 41, RFC 894, April 1984.

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

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

[RFC2462] Thomson, S. and T. Narten, "IPv6 Stateless Address Autoconfiguration", RFC 2462, December 1998.

[RFC2464] Crawford, M., "Transmission of IPv6 Packets over Ethernet Networks", RFC 2464, December 1998.

[RFC3315] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, July 2003.

[RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU Discovery", RFC 4821, March 2007.

9.2. Informative References

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

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. About of larger packets

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

1. Increased delay and jitter

2. Increased reliance on path MTU discovery

3. Increased packet loss through bit errors

4. Increased risk of undetected bit errors

B.1. 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.

B.2. 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 involve undue risks.

B.3. 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.4. 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 376 and 11454 bytes long (including) have a Hamming distance of 3. (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 or two 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 3 for a large range of packet sizes, this means that the risk of undetected errors goes up with the square of the packet length, but goes down with the third 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 11454 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 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.5. 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.6. IEEE 802.3 compatibility

According to the IEEE 802.3 standard, 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.

B.7. 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, and may also be of interest to ISPs.

Author's Address

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

Email: iljitsch@muada.com