Network Working Group
Request for Comments: 4459
Category: Informational

P. Savola
CSC/FUNET

April 2006

MTU and Fragmentation Issues with In-the-Network Tunneling

Status of This Memo

Copyright Notice

Abstract

Tunneling techniques such as IP-in-IP when deployed in the middle of the network, typically between routers, have certain issues regarding how large packets can be handled: whether such packets would be fragmented and reassembled (and how), whether Path MTU Discovery would be used, or how this scenario could be operationally avoided. This memo justifies why this is a common, non-trivial problem, and goes on to describe the different solutions and their characteristics at some length.

Table of Contents

1. Introduction

A large number of ways to encapsulate datagrams in other packets, i.e., tunneling mechanisms, have been specified over the years: for example, IP-in-IP (e.g., [1] [2], [3]), Generic Routing Encapsulation (GRE) [4], Layer 2 Tunneling Protocol (L2TP) [5], or IP Security (IPsec) [6] in tunnel mode -- any of which might run on top of IPv4, IPv6, or some other protocol and carrying the same or a different protocol.

All of these can be run so that the endpoints of the inner protocol are co-located with the endpoints of the outer protocol; in a typical scenario, this would correspond to "host-to-host" tunneling. It is also possible to have one set of endpoints co-located, i.e., host-to-router or router-to-host tunneling. Finally, many of these mechanisms are also employed between the routers for all or a part of the traffic that passes between them, resulting in router-to-router tunneling.

All these protocols and scenarios have one issue in common: how does the source select the maximum packet size so that the packets will fit, even encapsulated, in the smallest Maximum Transmission Unit (MTU) of the traversed path in the network; and if you cannot affect the packet sizes, what do you do to be able to encapsulate them in any case? The four main solutions are as follows (these will be elaborated in Section 3):

1. Fragmenting all too big encapsulated packets to fit in the paths, and reassembling them at the tunnel endpoints.

2. Signal to all the sources whose traffic must be encapsulated, and is larger than fits, to send smaller packets, e.g., using Path MTU Discovery (PMTUD)[7][8].

3. Ensure that in the specific environment, the encapsulated packets will fit in all the paths in the network, e.g., by using MTU bigger than 1500 in the backbone used for encapsulation.

4. Fragmenting the original too big packets so that their fragments will fit, even encapsulated, in the paths, and reassembling them at the destination nodes. Note that this approach is only available for IPv4 under certain assumptions (see Section 3.4).

It is also common to run multiple layers of encapsulation, for example, GRE or L2TP over IPsec; with nested tunnels in the network, the tunnel endpoints can be the same or different, and both the inner and outer tunnels may have different MTU handling strategies. In

particular, signalling may be a scalable option for the outer tunnel or tunnels if the number of innermost tunnel endpoints is limited.

The tunneling packet size issues are relatively straightforward in host-to-host tunneling or host-to-router tunneling where Path MTU Discovery only needs to signal to one source node. The issues are significantly more difficult in router-to-router and certain router-to-host scenarios, which are the focus of this memo.

It is worth noting that most of this discussion applies to a more generic case, where there exists a link with a lower MTU in the path. A concrete and widely deployed example of this is the usage of PPP over Ethernet (PPPoE) [11] at the customers' access link. These lower-MTU links, and particularly PPPoE links, are typically not deployed in topologies where fragmentation and reassembly might be unfeasible (e.g., a backbone), so this may be a slightly easier problem. However, this more generic case is considered out of scope of this memo.

There are also known challenges in specifying and implementing a mechanism that would be used at the tunnel endpoint to obtain the best suitable packet size to use for encapsulation: if a static value is chosen, a lot of fragmentation might end up being performed. On the other hand, if PMTUD is used, the implementation would need to update the discovered interface MTU based on the ICMP Packet Too Big messages and originate ICMP Packet Too Big message(s) back to the source(s) of the encapsulated packets; this also assumes that sufficient data has been piggybacked on the ICMP messages (beyond the required 64 bits after the IPv4 header). We'll discuss using PMTUD to signal the sources briefly in Section 3.2, but in-depth specification and analysis are described elsewhere (e.g., in [4] and [2]) and are out of scope of this memo.

Section 2 includes a problem statement, section 3 describes the different solutions with their drawbacks and advantages, and section 4 presents conclusions.

2. Problem Statement

It is worth considering why exactly this is considered a problem.

It is possible to fix all the packet size issues using solution 1, fragmenting the resulting encapsulated packet, and reassembling it by the tunnel endpoint. However, this is considered problematic for at least three reasons, as described in Section 3.1.

Therefore, it is desirable to avoid fragmentation and reassembly if possible. On the other hand, the other solutions may not be

practical either: especially in router-to-router or router-to-host tunneling, Path MTU Discovery might be very disadvantageous -- consider the case where a backbone router would send ICMP Packet Too Big messages to every source that would try to send packets through it. Fragmenting before encapsulation is also not available in IPv6, and not available when the Don't Fragment (DF) bit has been set (see Section 3.4 for more). Ensuring a high enough MTU so encapsulation is always possible is of course a valid approach, but requires careful operational planning, and may not be a feasible assumption for implementors.

This yields that there is no trivial solution to this problem, and it needs to be further explored to consider the trade offs, as is done in this memo.

3. Description of Solutions

This section describes the potential solutions in a bit more detail.

3.1. Fragmentation and Reassembly by the Tunnel Endpoints

The seemingly simplest solution to tunneling packet size issues is fragmentation of the outer packet by the encapsulator and reassembly by the decapsulator. However, this is highly problematic for at least three reasons:

- Fragmentation causes overhead: every fragment requires the IP header (20 or 40 bytes), and with IPv6, an additional 8 bytes for the Fragment Header.

- Fragmentation and reassembly require computation: splitting datagrams to fragments is a non-trivial procedure, and so is their reassembly. For example, software router forwarding implementations may not be able to perform these operations at line rate.

- At the time of reassembly, all the information (i.e., all the fragments) is normally not available; when the first fragment arrives to be reassembled, a buffer of the maximum possible size may have to be allocated because the total length of the reassembled datagram is not known at that time. Furthermore, as fragments might get lost, or be reordered or delayed, the reassembly engine has to wait with the partial packet for some time (e.g., 60 seconds [9]). When this would have to be done at the line rate, with, for example 10 Gbit/s speed, the length of the buffers that reassembly might require would be prohibitive.

When examining router-to-router tunneling, the third problem is likely the worst; certainly, a hardware computation and implementation requirement would also be significant, but not all that difficult in the end -- and the link capacity wasted in the backbones by additional overhead might not be a huge problem either.

However, IPv4 identification header length is only 16 bits (compared to 32 bits in IPv6), and if a larger number of packets are being tunneled between two IP addresses, the ID is very likely to wrap and cause data misassociation. This reassembly wrongly combining data from two unrelated packets causes data integrity and potentially a confidentiality violation. This problem is further described in [12].

IPv6, and IPv4 with the DF bit set in the encapsulating header, allows the tunnel endpoints to optimize the tunnel MTU and minimize network-based reassembly. This also prevents fragmentation of the encapsulated packets on the tunnel path. If the IPv4 encapsulating header does not have the DF bit set, the tunnel endpoints will have to perform a significant amount of fragmentation and reassembly, while the use of PMTUD is minimized.

As Appendix A describes, the MTU of the tunnel is also a factor on which packets require fragmentation and reassembly; the worst case occurs if the tunnel MTU is "infinite" or equal to the physical interface MTUs.

So, if reassembly could be made to work sufficiently reliably, this would be one acceptable fallback solution but only for IPv6.

3.2. Signalling the Lower MTU to the Sources

Another approach is to use techniques like Path MTU Discovery (or potentially a future derivative [13]) to signal to the sources whose packets will be encapsulated in the network to send smaller packets so that they can be encapsulated; in particular, when done on routers, this includes two separable functions:

a. Forwarding behaviour: when forwarding packets, if the IPv4-only DF bit is set, the router sends an ICMP Packet Too Big message to the source if the MTU of the egress link is too small.

b. Router’s "host" behaviour: when the router receives an ICMP Packet Too Big message related to a tunnel, it (1) adjusts the tunnel MTU, and (2) originates an ICMP Packet Too Big message to the source address of the encapsulated packet. (2) can be done either immediately or by waiting for the next packet to trigger an ICMP; the former minimizes the packet loss due to MTU changes.

Note that this only works if the MTU of the tunnel is of reasonable size, and not, for example, 64 kilobytes: see Appendix A for more.

This approach would presuppose that PMTUD works. While it is currently working for IPv6, and critical for its operation, there is ample evidence that in IPv4, PMTUD is far from reliable due to, for example firewalls and other boxes being configured to inappropriately drop all the ICMP packets [14], or software bugs rendering PMTUD inoperational.

Furthermore, there are two scenarios where signalling from the network would be highly undesirable. The first is when the encapsulation would be done in such a prominent place in the network that a very large number of sources would need to be signalled with this information (possibly even multiple times, depending on how long they keep their PMTUD state). The second is when the encapsulation is done for passive monitoring purposes (network management, lawful interception, etc.) -- when it's critical that the sources whose traffic is being encapsulated are not aware of this happening.

When desiring to avoid fragmentation, IPv4 requires one of two alternatives [1]: copy the DF bit from the inner packets to the encapsulating header, or always set the DF bit of the outer header. The latter is better, especially in controlled environments, because it forces PMTUD to converge immediately.

A related technique, which works with TCP under specific scenarios only, is so-called "MSS clamping". With that technique or rather a "hack", the TCP packets' Maximum Segment Size (MSS) is reduced by tunnel endpoints so that the TCP connection automatically restricts itself to the maximum available packet size. Obviously, this does not work for UDP or other protocols that have no MSS. This approach is most applicable and used with PPPoE, but could be applied otherwise as well; the approach also assumes that all the traffic goes through tunnel endpoints that do MSS clamping -- this is trivial for the single-homed access links, but could be a challenge otherwise.

A new approach to PMTUD is in the works [13], but it is uncertain whether that would fix the problems -- at least not the passive monitoring requirements.

3.3. Encapsulate Only When There is Free MTU

The third approach is an operational one, depending on the environment where encapsulation and decapsulation are being performed. That is, if an ISP would deploy tunneling in its backbone, which would consist only of links supporting high MTUs

(e.g., Gigabit Ethernet or SDH/SONET), but all its customers and peers would have a lower MTU (e.g., 1500, or the backbone MTU minus the encapsulation overhead), this would imply that no packets (with the encapsulation overhead added) would have a larger MTU than the "backbone MTU", and all the encapsulated packets would always fit MTU-wise in the backbone links.

This approach is highly assumptive of the deployment scenario. It may be desirable to build a tunnel to/from another ISP, for example, where this might no longer hold; or there might be links in the network that cannot support the higher MTUs to satisfy the tunneling requirements; or the tunnel might be set up directly between the customer and the ISP, in which case fragmentation would occur, with tunneled fragments terminating on the ISP and thus requiring reassembly capability from the ISP's equipment.

To restate, this approach can only be considered when tunneling is done inside a part of specific kind of ISP's own network, not, for example, transiting an ISP.

Another, related approach might be having the sources use only a low enough MTU that would fit in all the physical MTUs; for example, IPv6 specifies the minimum MTU of 1280 bytes. For example, if all the sources whose traffic would be encapsulated would use this as the maximum packet size, there would probably always be enough free MTU for encapsulation in the network. However, this is not the case today, and it would be completely unrealistic to assume that this kind of approach could be made to work in general.

It is worth remembering that while the IPv6 minimum MTU is 1280 bytes [10], there are scenarios where the tunnel implementation must implement fragmentation and reassembly [3]: for example, when having an IPv6-in-IPv6 tunnel on top of a physical interface with an MTU of 1280 bytes, or when having two layers of IPv6 tunneling. This can only be avoided by ensuring that links on top of which IPv6 is being tunneled have a somewhat larger MTU (e.g., 40 bytes) than 1280 bytes. This conclusion can be generalized: because IP can be tunneled on top of IP, no single minimum or maximum MTU can be found such that fragmentation or signalling to the sources would never be needed.

All in all, while in certain operational environments it might be possible to avoid any problems by deployment choices, or limiting the MTU that the sources use, this is probably not a sufficiently good general solution for the equipment vendors. Other solutions must also be provided.

3.4. Fragmentation of the Inner Packet

A final possibility is fragmenting the inner packet, before encapsulation, in such a manner that the encapsulated packet fits in the tunnel's path MTU (discovered using PMTUD). However, one should note that only IPv4 supports this "in-flight" fragmentation; furthermore, it isn't allowed for packets where the Don't Fragment bit has been set. Even if one could ignore IPv6 completely, so many IPv4 host stacks send packets with the DF bit set that this would seem unfeasible.

However, there are existing implementations that violate the standard that:

- discard too big packets with the DF bit not set instead of fragmenting them (this is rare);
- ignore the DF bit completely, for all or specified interfaces; or
- clear the DF bit before encapsulation, in the egress of configured interfaces. This is typically done for all the traffic, not just too big packets (allowing configuring this is common).

This is non-compliant behaviour, but there are certainly uses for it, especially in certain tightly controlled passive monitoring scenarios, and it has potential for more generic applicability as well, to work around PMTUD issues.

Clearing the DF bit effectively disables the sender's PMTUD for the path beyond the tunnel. This may result in fragmentation later in the network, but as the packets have already been fragmented prior to encapsulation, this fragmentation later on does not make matters significantly worse.

As this is an implemented and desired (by some) behaviour, the full impacts e.g., for the functioning of PMTUD (for example) should be analyzed, and the use of fragmentation-related IPv4 bits should be re-evaluated.

In summary, this approach provides a relatively easy fix for IPv4 problems, with potential for causing problems for PMTUD; as this would not work with IPv6, it could not be considered a generic solution.

4. Conclusions

Fragmentation and reassembly by the tunnel endpoints are a clear and simple solution to the problem, but the hardware reassembly when the packets get lost may face significant implementation challenges that may be insurmountable. This approach does not seem feasible, especially for IPv4 with high data rates due to problems with wrapping the fragment identification field [12]. Constant wrapping may occur when the data rate is in the order of MB/s for IPv4 and in the order of dozens of GB/s for IPv6. However, this reassembly approach is probably not a problem for passive monitoring applications.

PMTUD techniques, at least at the moment and especially for IPv4, appear to be too unreliable or unscalable to be used in the backbones. It is an open question whether a future solution might work better in this aspect.

It is clear that in some environments the operational approach to the problem, ensuring that fragmentation is never necessary by keeping higher MTUs in the networks where encapsulated packets traverse, is sufficient. But this is unlikely to be enough in general, and for vendors that may not be able to make assumptions about the operators' deployments.

Fragmentation of the inner packet is only possible with IPv4, and is sufficient only if standards-incompliant behaviour, with potential for bad side-effects (e.g., for PMTUD), is adopted. It should not be used if there are alternatives; fragmentation of the outer packet seems a better option for passive monitoring.

However, if reassembly in the network must be avoided, there are basically two possibilities:

1. For IPv6, use ICMP signalling or operational methods.

2. For IPv4, packets for which the DF bit is not set can be fragmented before encapsulation (and the encapsulating header would have the DF bit set); packets whose DF bit is set would need to get the DF bit cleared (though this is non-compliant). This also minimizes the need for (unreliable) Internet-wide PMTUD.

An interesting thing to explicitly note is that when tunneling is done in a high-speed backbone, typically one may be able to make assumptions on the environment; however, when reassembly is not performed in such a network, it might be done in software or with lower requirements, and there exists either a reassembly

implementation using PMTUD or using a separate approach for passive monitoring -- so this might not be a real problem.

In consequence, the critical questions at this point appear to be 1) whether a higher MTU can be assumed in the high-speed networks that deploy tunneling, and 2) whether "slower-speed" networks could cope with a software-based reassembly, a less capable hardware-based reassembly, or the other workarounds. An important future task would be analyzing the observed incompliant behaviour about the DF bit to note whether it has any unanticipated drawbacks.

5. Security Considerations

This document describes different issues with packet sizes and in-the-network tunneling; this does not have security considerations on its own.

However, different solutions might have characteristics that may make them more susceptible to attacks -- for example, a router-based fragment reassembly could easily lead to (reassembly) buffer memory exhaustion if the attacker sends a sufficient number of fragments without sending all of them, so that the reassembly would be stalled until a timeout; these and other fragment attacks (e.g., [15]) have already been used against, for example, firewalls and host stacks, and need to be taken into consideration in the implementations.

It is worth considering the cryptographic expense (which is typically more significant than the reassembly, if done in software) with fragmentation of the inner or outer packet. If an outer fragment goes missing, no cryptographic operations have been yet performed; if an inner fragment goes missing, cryptographic operations have already been performed. Therefore, which of these approaches is preferable also depends on whether cryptography or reassembly is already provided in hardware; for high-speed routers, at least, one should be able to assume that if it is performing relatively heavy cryptography, hardware support is already required.

The solutions using PMTUD (and consequently ICMP) will also need to take into account the attacks using ICMP. In particular, an attacker could send ICMP Packet Too Big messages indicating a very low MTU to reduce the throughput and/or as a fragmentation/reassembly denial-of-service attack. This attack has been described in the context of TCP in [16].

6. Acknowledgements

While the topic is far from new, recent discussions with W. Mark Townsley on L2TP fragmentation issues caused the author to sit down and write up the issues in general. Michael Richardson and Mika Joutsenvirta provided useful feedback on the first version. When soliciting comments from the NANOG list, Carsten Bormann, Kevin Miller, Warren Kumari, Iljitsch van Beijnum, Alok Dube, and Stephen J. Wilcox provided useful feedback. Later, Carlos Pignataro provided excellent input, helping to improve the document. Joe Touch also provided input on the memo.

7. References

7.1. Normative References

[1] Perkins, C., "IP Encapsulation within IP", RFC 2003, October 1996.

[2] Nordmark, E. and R. Gilligan, "Basic Transition Mechanisms for IPv6 Hosts and Routers", RFC 4213, October 2005.

[3] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, December 1998.

[4] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, March 2000.

[5] Lau, J., Townsley, M., and I. Goyret, "Layer Two Tunneling Protocol - Version 3 (L2TPv3)", RFC 3931, March 2005.

[6] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

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

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

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

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

7.2. Informative References

[11] Mamakos, L., Lidl, K., Evarts, J., Carrel, D., Simone, D., and R. Wheeler, "A Method for Transmitting PPP Over Ethernet (PPPoE)", RFC 2516, February 1999.

[12] Mathis, M., "Fragmentation Considered Very Harmful", Work in Progress, July 2004.

[13] Mathis, M. and J. Heffner, "Path MTU Discovery", Work in Progress, March 2006.

[14] Medina, A., Allman, M., and S. Floyd, "Measuring the Evolution of Transport Protocols in the Internet", Computer Communications Review, Apr 2005, <http://www.icir.org/tbit/>.

[15] Miller, I., "Protection Against a Variant of the Tiny Fragment Attack (RFC 1858)", RFC 3128, June 2001.

[16] Gont, F., "ICMP attacks against TCP", Work in Progress, February 2006.

Appendix A. MTU of the Tunnel

Different tunneling mechanisms may treat the tunnel links as having different kinds of MTU values. Some might use the same default MTU as for other interfaces; some others might use the default MTU minus the expected IP overhead (e.g., 20, 28, or 40 bytes); some others might even treat the tunnel as having an "infinite MTU", e.g., 64 kilobytes.

As [2] describes, having an infinite MTU, i.e., always fragmenting the outer packet (and never the inner packet) and never performing PMTUD for the tunnel path, is a very bad idea, especially in host-to-router scenarios. (It could be argued that if the nodes are sure that this is a host-to-host tunnel, a larger MTU might make sense if fragmentation and reassembly are more efficient than just sending properly sized packets -- but this seems like a stretch.)

Author's Address

Pekka Savola
CSC/FUNET
Espoo
Finland

EMail: psavola@funet.fi

Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).