

Packet Fragmentation in IPv6 over IPv4 Tunnels

Xin Liu
Department of Electronic Engineering
Tsinghua Univ.
lx@ns.6test.edu.cn

Xing Li
Department of Electronic Engineering
Tsinghua Univ.
xing@cernet.edu.cn

ABSTRACT

Nowadays IPv6 over IPv4 tunnels are widely used to form the global IPv6 Internet. This paper analyzes a packet fragmentation problem in IPv6 over IPv4 tunnels due to the MTU difference between IPv6 and IPv4 layers, which would greatly degrade the performance of the tunnels if ever happened. It also demonstrates an ICMP based attack that could induce the problem and gives some advice on how to deal with such attacks.

Keywords

IPv6, MTU, Fragmentation, Tunnel, Security

1. INTRODUCTION

IPv6 is the next generation IP, which is incompatible with IPv4. Nowadays IPv6 deployment has come to the stage that all over the world there are many IPv6 enabled networks, small or large, which are separated from each other by the intermediate routers that know only of IPv4. These networks are called *native IPv6 networks*. They are interconnected by IPv6 over IPv4 tunnels and then become the global IPv6 Internet. An IPv6 over IPv4 tunnel is a virtual link at the IPv6 layer, through which IPv6 packets could be transmitted by being encapsulated in IPv4 packets. This tunneling technique is demonstrated in Figure 1.

Today IPv6 over IPv4 tunnels are widely used to connect large regional IPv6 networks, because it is relatively difficult to build an international or cross-continent native IPv6 network. This makes the characteristics of IPv6 over IPv4 tunnels very important to the performance of the global IPv6 Internet.

MTU is an essential concept in any packet-switching system. Link MTU is the upper limit to size of the packets that could traverse a link, and path MTU is the size of the largest packet that could go unfragmented from one node to another. Path MTU equals to the smallest link MTU on the

Table 1: Various MTUs

| MTU | Network |
|-------|--------------------------------|
| 65535 | Hyperchannel |
| 17914 | 16Mb IBM Token Ring |
| 8166 | IEEE 802.4 |
| 4464 | IEEE 802.5 (4Mb max) |
| 4352 | FDDI (Revised) |
| 2048 | Wideband Network |
| 2002 | IEEE 802.4 (4Mb recommended) |
| 1536 | Experimental Ethernet |
| 1500 | Ethernet |
| 1492 | IEEE 802.3 |
| 1006 | ARPANET |
| 576 | X.25 Networks |
| 544 | DEC IP Portal |
| 512 | NETBIOS |
| 508 | ARCNET |
| 296 | Point-to-Point (low delay) |
| 68 | Official Minimum MTU |

path from the source to the destination. Link MTU varies in the Internet due to different networking technologies beneath the IP layer. Table 1 is a list of different physical networks and their link MTUs.

In the IPv4 specification [1] it is stated that every internet module must be able to forward a datagram of 68 octets without further fragmentation, so the minimum MTU at the IPv4 layer is 68 bytes. But in the IPv6 specification [2], it is required that every link in the internet have an MTU of 1280 octets or greater. The difference between the MTUs of IPv4 and IPv6 could possibly induce a packet fragmentation problem in IPv6 over IPv4 tunnels.

2. PACKET FRAGMENTATION PROBLEM

Packet fragmentation caused by IPv4 and IPv6 MTU difference may impose unfavorable consequences, and it is very difficult to keep it from happening in some circumstances.

2.1 Problem Description

The situation is depicted in Figure 2. Router_A and Router_C are IPv6 and IPv4 dual stack routers. They belong to different native IPv6 networks, and they are connected by Router_B, which has only IPv4 protocol stack and thus understands only IPv4. An IPv6 over IPv4 tunnel is established between Router_A and Router_C, which is represented by a dashed line in the figure.

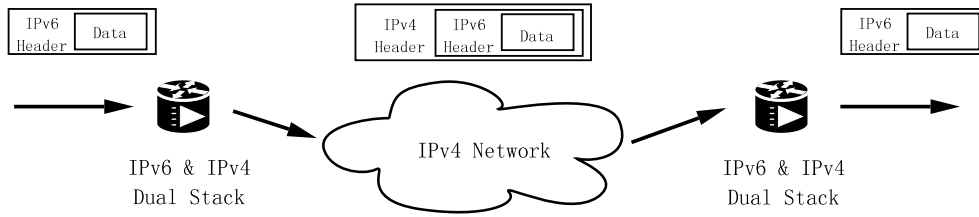


Figure 1: IPv6 over IPv4 Tunnel

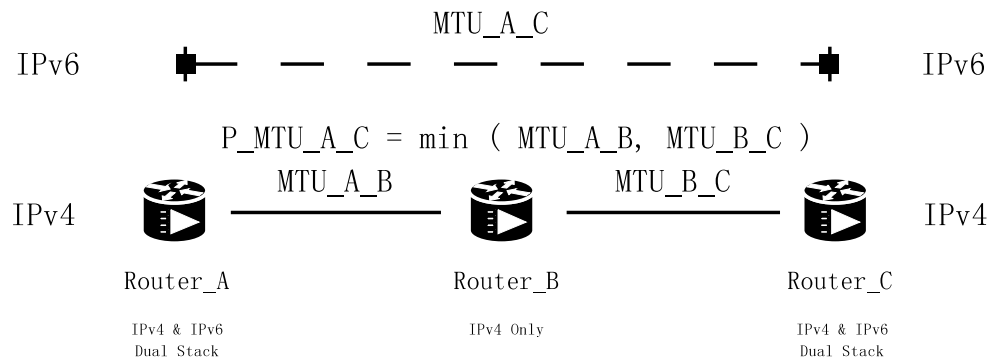


Figure 2: IPv6 over IPv4 Tunnel and various MTUs

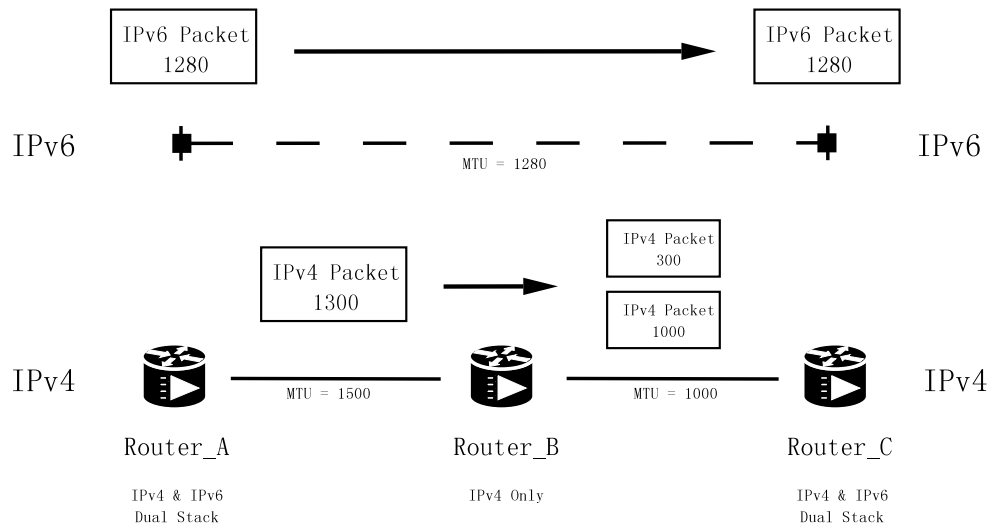


Figure 3: Packet Fragmentation

At the IPv4 layer, $MTU_{A,B}$ is the link MTU between Router_A and Router_B, and $MTU_{B,C}$ is the link MTU between Router_B and Router_C. The path MTU from Router_A to Router_C is $P_MTU_{A,C}$, which equals to the smaller one of $MTU_{A,B}$ and $MTU_{B,C}$. At the IPv6 layer, the tunnel between Router_A and Router_C is a virtual link that is conceptually no different from any physical link, so it has its own link MTU, which is $MTU_{A,C}$.

Now a problem arises from the difference between $MTU_{A,C}$ and $P_MTU_{A,C}$. In order to avoid any packet fragmentation, $MTU_{A,C}$ should be at least 20 bytes smaller than $P_MTU_{A,C}$, because for transmission at the IPv4 layer each IPv6 packet flowing from Router_A to Router_C is prepended with an IPv4 packet header that is 20 bytes long at minimum. If $MTU_{A,C}$ is too large, for instance if $MTU_{A,C}$ is 1280 bytes and $P_MTU_{A,C}$ is 1000 bytes, an IPv6 packet carrying 1280 bytes from Router_A to Router_C will have to be split into two packets at the IPv4 layer, as shown in Figure 3.

2.2 Consequences

This kind of packet fragmentation is especially harmful to the network. Fragmentation brings about many problems [3], including inefficient use of resources, degraded performance and difficulty in efficient reassembling. In the real world IPv6 over IPv4 tunnels are generally connecting remotely stationed IPv6 routers, which means that there are likely to be much more IPv4-only routers between Router_A and Router_C in Figure 2. Therefore, when the packet fragmentation happens, delay, duplication, loss and out-of-sequence of the IPv4 packet fragments from Router_A to Router_C are particularly easy to occur, and Router_C would be heavily burdened with the packet reassembling task. The tunnel between Router_A and Router_C, as a virtual link at the IPv6 layer, would be greatly affected at its characteristics: the average delay would increase, the standard deviation of the delay would enlarge, and the IPv6 packet loss would rise. All these deteriorations of link quality would produce a severe impact on some applications in the IPv6 network such as real-time multimedia streaming.

To make it worse, nothing at the IPv6 layer could be able to sense the occurrence of the packet fragmentation at the IPv4 layer, because as demonstrated in Figure 3 the IPv6 packet is seen by the IPv6 protocol stack to traverse unfragmented. So nothing could be done in the IPv6 world to reduce the impact of the packet fragmentation: in the case of Figure 3, all the IPv6 applications would be happy at sending out IPv6 packets of 1280 bytes instead of making them 980 bytes long, the latter of which could completely eliminate the underlying IPv4 packet fragmentation.

2.3 Solutions

Now we come to the question: is this packet fragmentation problem avoidable? It depends. In some circumstances it is completely unavoidable, while in others it might be eliminated with some sacrifice. Let us take Figure 2 as the example. To make it more realistic, let us further suppose that Router_A and Router_C are far away from each other with a lot of IPv4-only routers between them, and $P_MTU_{A,C}$ is the path MTU from Router_A to Router_C instead of the smaller one of $MTU_{A,B}$ and $MTU_{B,C}$.

First of all, the minimum link MTU of IPv6 is 1280 bytes, while the same value of IPv4 is only 68 bytes and there are a number of physical networks whose MTUs are smaller than 1300 bytes according to Table 1. As long as $P_MTU_{A,C}$ is smaller than 1300 bytes, packet fragmentation is unavoidable. If this ever happens, the only thing we can do is to design a special fragmentation procedure that tries to reduce the impact.

Now let us suppose $P_MTU_{A,C}$ is always larger than 1300 bytes. With this assumption the packet fragmentation could be eliminated, but price has to be paid, which might or might not be acceptable. Since an IPv6 over IPv4 tunnel is a virtual link, there is no physical limitation to its link MTU at the IPv6 layer, thus in theory $MTU_{A,C}$ could be set to any value.

1. The most simple solution is to always set $MTU_{A,C}$ to 1280 bytes. This is how FreeBSD-4.x deals with tunnel MTU. It surely promises freedom from packet fragmentation because $MTU_{A,C}$ is small enough. However, this solution wastes the capacity of the underlying physical networks, because larger MTU means better performance [7]. The waste is especially noticeable when all the IPv4 routers are connected by SDH links where MTU could be around 9K bytes. Sometimes this sacrifice in performance might not be affordable.
2. A more optimal solution is to associate $MTU_{A,C}$ with the value of $P_MTU_{A,C}$. To make sure packet fragmentation would not happen, $MTU_{A,C}$ could be set to at most $P_MTU_{A,C}$ minus 20 bytes. Things are done this way in Linux-2.2, Linux-2.4 and Linux-2.6. However, $P_MTU_{A,C}$ is only available via measurement, thus a path MTU discovery scheme has to be adopted. It is difficult to come up with an efficient and trouble-free path MTU discovery algorithm, because $P_MTU_{A,C}$ is not a constant due to the dynamic topology of the Internet, and it would even be likely to change at any time if the IPv4 network between Router_A and Router_C is sophisticated enough. The price of adding a not-so-good path MTU discovery scheme might not be acceptable for some systems considering its resource consumption and other problems. Actually the path MTU discovery scheme in Linux-2.2, Linux-2.4 and Linux-2.6 is vulnerable to some ICMP based attacks, which will be detailed in the next section.

3. PATH MTU DISCOVERY IN LINUX AND ITS FLAWS

Path MTU discovery is widely used in Linux-2.2, Linux-2.4 and Linux-2.6. However, there are some security flaws in it that could be used to trigger the analyzed packet fragmentation problem.

For easier notation, in the remaining parts of this paper "Linux" will be used as an abbreviation for Linux-2.2, Linux-2.4 and Linux-2.6. Earlier versions of Linux kernel are not studied because they are seldom used now.

3.1 Problem Description

Path MTU discovery has been an issue in the Internet for quite a long time, because as early as 1987 it had already been indicated that IP fragmentation is harmful [3]. A path MTU discovery scheme was proposed in [4], which is both straight-forward and simple, and thus widely used in the Internet. Precisely speaking it is designed for end-to-end communication protocols such as UDP and TCP, but its idea of adjusting path MTU on receiving *TooBig* ICMP messages is adopted at the IP layer in many OSes. In Linux, IPv4 path MTU cache is part of the IPv4 routing cache, and the cached path MTU values play important roles in various parts of the IPv4 protocol stack including TCP and UDP initialization, packet forwarding and packet output.

IPv4 Path MTU discovery works this way in Linux. There is a field for storing corresponding path MTU in a routing cache entry. When a routing cache entry is created, its path MTU field is initialized with the link MTU of the outgoing network interface. When a *TooBig* ICMP message is received, the routing cache entry corresponding to the included IPv4 packet header will be located, and its path MTU field will be updated according to the MTU information contained in the ICMP message.

An attacker could reduce the path MTU values stored in the routing cache by sending in false *TooBig* ICMP messages, and this could lead to the analyzed packet fragmentation problem. Let us still use Figure 2 as a sample network topology. Let us further suppose that Router_A is running Linux on it and MTU_A_B and MTU_B_C are both 1500 bytes. When the first IPv6 packet is forwarded from Router_A to Router_C, an IPv4 routing cache entry will be created on Router_A whose destination is Router_C and whose path MTU field is initialized with MTU_A_B, i.e. 1500 bytes. According to the definition this path MTU field stores P_MTU_A_C. Now anyone on the Internet could send an IPv4 *TooBig* ICMP message to Router_A, saying that an IPv4 packet from Router_A to Router_C could not be forwarded because its size exceeds the link MTU that is 1000 bytes. Router_A receives this ICMP message and reduces P_MTU_A_C to 1000 bytes. Then MTU_A_C will be updated to P_MTU_A_C minus 20 bytes, but considering that IPv6 link MTU should not be smaller than 1280 bytes, the system could only set MTU_A_C to 1280 bytes. Now the situation becomes similar to Figure 3 except that IPv4 packet fragmentation is done by Router_A, not Router_B. The analyzed problem occurs.

This ICMP based attack is very difficult to deal with for several reasons.

- IPv4 *TooBig* ICMP messages could not be filtered off by firewalls, because they are essential for TCP path MTU discovery [5] [6].
- It is impossible for Router_A to distinguish normal and false IPv4 *TooBig* ICMP messages.
- It is difficult to identify this kind of attack. By default the value stored in a path MTU field expires in several minutes, so an attacker could send out false messages between relatively long intervals, making them more undistinguishable from normal *TooBig* ICMP messages.

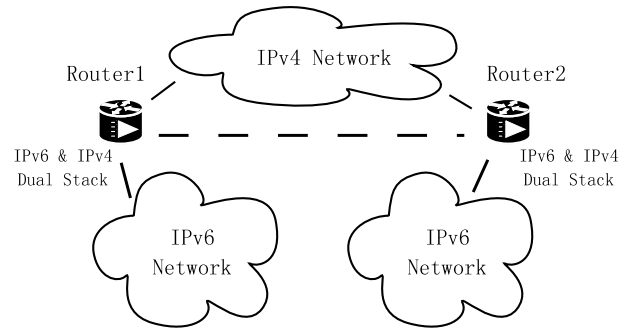


Figure 4: ICMP Attack Environment

- It is very difficult to trace the attack source, because an attacker could put random numbers in the source IPv4 address field of the false messages.

We have successfully performed the attack in an experimental environment as depicted in Figure 4. After a false IPv4 *TooBig* ICMP message is sent to Router1, the analyzed packet fragmentation is immediately observed on Router1. Without further attacks, the reduced path MTU returns to normal and the fragmentation stops in several minutes.

Attention has to be paid to a few details in order to make the attack successful.

- The IPv4 header included as data in the false ICMP message should be constructed in a logical manner because some simple checks will be made on it when the attacked router receives the message.
- An IPv6 packet has to be forwarded from Router1 to Router2 before the attack in order to have the necessary routing cache entry created on Router1.

3.2 Solutions

Two solutions are available to defend against the attack without modifying the Linux kernel, but neither of them is perfect.

- There is a configurable threshold for IPv4 path MTU, which could be manipulated via the file `/proc/sys/net/ipv4/route/min_pmtu`. None of the IPv4 path MTU values used in the system could be smaller than it. Its default value is 552 bytes, and if we set it to 1300 bytes, the attack could no longer induce the packet fragmentation problem. Even if the actual path MTU between the tunnel endpoints is smaller than 1300 bytes, this setting causes no additional problems because packet fragmentation is completely unavoidable under this condition according to our previous analysis. However, this threshold is a system-wide parameter that influence not only IPv6 over IPv4 tunnels, so for the sake of other parts of the system it might not be able to go as large as 1300 bytes.
- Disabling IPv4 path MTU discovery by manipulating the file `/proc/sys/net/ipv4/ip_no_pmtu_disc` could

surely make the attack useless. Again, this is also a system-wide parameter and it might not be disabled for other parts of the system to function properly.

4. CONCLUSION

In this paper we located a packet fragmentation problem in IPv6 over IPv4 tunneling which is caused by the MTU difference between IPv6 and IPv4 layers. We pointed out the consequences of the problem and discussed about the possibility of avoiding the problem.

We also found that the packet fragmentation problem may be triggered by attackers even when it would not automatically happen, and we successfully performed this kind of attack to our experimental Linux systems. Finally we suggested two dirty solutions for defending this kind of attack in Linux systems.

In the future we shall try to find a clean solution to deal with the attack for Linux systems. Moreover we shall investigate the main stream routers to check whether they are vulnerable to similar attacks and try to find the solutions if they are.

5. REFERENCES

- [1] Jon Postel, Internet Protocol (RFC791), 1981
- [2] S. Deering, R. Hinden, Internet Protocol Version 6 (RFC2460), 1998
- [3] Christopher A. Kent, Jeffrey C. Mogul, Fragmentation Considered Harmful, Proc. SIGCOMM '87 Workshop, p.390-401, Stowe, VT, 1987
- [4] J. Mogul, S. Deering, Path MTU Discovery (RFC1191), 1990
- [5] K. Lahey, TCP Problems with Path MTU Discovery (RFC2923), 2000
- [6] Richard van den Berg, Phil Dibowitz, Over-Zealous Security Administrators Are Breaking the Internet, Proc. of LISA 2002: 16th Systems Administration Conference, p.213-218, Berkeley, USA, 2002
- [7] Matt Mathis, Effect of MTU on TCP performance, <http://www.psc.edu/~rreddy/networking/mtu.html>, 2004