Avoiding IP Fragmentation at the Transport Layer of the OSI Reference Model

Delian Genkov, Raycho Ilarionov

Abstract: *The heterogeneous nature of the Internet communications is transparent for the end user. One of the factors determining the transit delays is the presence of fragmentation and reassembly of the IP datagrams. The modern recommendations regarding the IP fragmentation process is to avoid it. There are different possibilities regarding the OSI model layers to implement logic for avoiding fragmentation. The present study describes the possibility to use the built options of the Transport Layer for avoiding fragmentation.*

Key words: *IP, Fragmentation, Reassembly, OSI, TCP.*

INTRODUCTION

The IP protocol was designed for use on a wide variety of transmission links. Although the maximum length of an IP datagram is 64K, most transmission links enforce a smaller maximum packet length limit, called a Maximum Transmission Unit (MTU). The value of the MTU depends on the type of the transmission link. The design of IP accommodates MTU differences by allowing routers to fragment IP datagrams as necessary. The receiving station is responsible for reassembling the fragments back into the original full size IP datagram.

IP fragmentation involves breaking a datagram into a number of pieces that can be reassembled later. The IP source, destination, identification, total length, and fragment offset fields, along with the "more fragments" and "don't fragment" flags in the IP header, are used for IP fragmentation and reassembly. For more information about the mechanics of IP fragmentation and reassembly, please see RFC 791[1].

There are several issues that make IP fragmentation undesirable. There is a small increase in CPU and memory overhead to fragment an IP datagram. This holds true for the sender as well as for a router in the path between a sender and a receiver. Creating fragments simply involves creating fragment headers and copying the original datagram into the fragments. This can be done fairly efficiently because all the information needed to create the fragments is immediately available.

Fragmentation causes more overhead for the receiver when reassembling the fragments because the receiver must allocate memory for the arriving fragments and coalesce them back into one datagram after all of the fragments are received. Reassembly on a host is not considered a problem because the host has the time and memory resources to devote to this task.

But, reassembly is very inefficient on a router whose primary job is to forward packets as quickly as possible. A router is not designed to hold on to packets for any length of time. Also a router doing reassembly chooses the largest buffer available (18K) with which to work because it has no way of knowing the size of the original IP packet until the last fragment is received.

Another fragmentation issue involves handling dropped fragments. If one fragment of an IP datagram is dropped, then the entire original IP datagram must be resent, and it will also be fragmented. You see an example of this with Network File System (NFS). NFS, by default, has a read and write block size of 8192, so a NFS IP/UDP datagram will be approximately 8500 bytes (including NFS, UDP, and IP headers). A sending station connected to an Ethernet (MTU 1500) will have to fragment the 8500 byte datagram into six pieces; five 1500 byte fragments and one 1100 byte fragment. If any of the six fragments is dropped because of a congested link, the complete original datagram will have to be retransmitted, which means that six more fragments will have to be created. If

this link drops one in six packets, then the odds are low that any NFS data can be transferred over this link, since at least one IP fragment would be dropped from each NFS 8500 byte original IP datagram.

Firewalls that filter or manipulate packets based on Layer 4 (L4) through Layer 7 (L7) information in the packet may have trouble processing IP fragments correctly. If the IP fragments are out of order, a firewall may block the non−initial fragments because they do not carry the information that would match the packet filter. This would mean that the original IP datagram could not be reassembled by the receiving host. If the firewall is configured to allow non−initial fragments with insufficient information to properly match the filter, then a non−initial fragment attack through the firewall could occur. Also, some network devices (such as Content Switch Engines) direct packets based on L4 through L7 information, and if a packet spans multiple fragments, then the device may have trouble enforcing its policies.

The modern recommended approaches regarding the IP datagrams fragmentation and reassembly process are to avoid fragmentation at any cost. That is because in most circumstances, the potential disadvantages of fragmentation far outweigh the expected advantages. Thus, hosts should avoid sending datagrams that are so large that they will be fragmented. Some of these approaches supposed always to send to the internet datagrams that are small enough to be fragmented. The official minimum MTU (Maximal Transmission Unit) which is required to be supported of all internet nodes, that is to transfer such datagrams without involving it in a fragmentation process is 576 bytes.

That's why many implementations avoids fragmentation without spending more resources to dealt with it, just with sending 576 bytes datagrams when the route is not on the local network and thus it may be involved in a fragmentation process.

This approach is quiet simple, but in most cases it causes an inefficient using of the network resources, because more of the modern networks, comprising the Internet are able to carry more than 576 bytes without fragmentation. That's why this and connected approaches are not taking in to consideration here, because the overall goal is to avoiding fragmentation, but still efficiently use the network's resources. That means to send datagrams with maximal or near to maximal size without to be fragmented. Most of these approaches assumes guessing or discovering this maximal size, called MTU below.

IP is layered protocol architecture, and fragmentation avoidance must be done at the right layer. It makes little sense to build redundant mechanisms into several layers if it is possible to do it once. [3]

AVOIDING IP FRAGMENTATION USING TCP MSS OPTION

The TCP Maximum Segment Size (MSS) defines the maximum amount of data that a host is willing to accept in a single TCP/IP datagram. This TCP/IP datagram may be fragmented at the IP layer. The MSS value is sent as a TCP header option only in TCP SYN segments. Each side of a TCP connection reports its MSS value to the other side. Contrary to popular belief, the MSS value is not negotiated between hosts. The sending host is required to limit the size of data in a single TCP segment to a value less than or equal to the MSS reported by the receiving host. [2]

Originally, MSS meant how big a buffer (greater than or equal to 65496K) was allocated on a receiving station to be able to store the TCP data contained within a single IP datagram. MSS was the maximum segment (chunk) of data that the TCP receiver was willing to accept. This TCP segment could be as large as 64K (the maximum IP datagram size) and it could be fragmented at the IP layer in order to be transmitted across the network to the receiving host. The receiving host would reassemble the IP datagram before it handed the complete TCP segment to the TCP layer. This process is shown at figure 1.

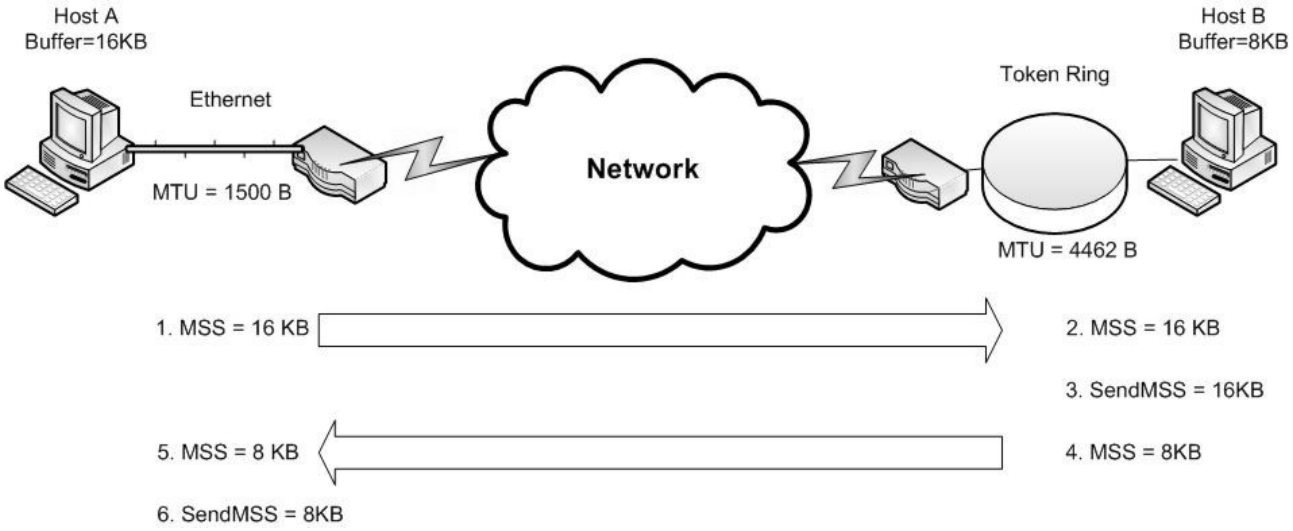

Fig. 1. Original Implementation of MSS

Host A has a buffer of 16K and Host B a buffer of 8K. They send and receive their MSS values and adjust their send MSS for sending data to each other. Notice that Host A and Host B will have to fragment the IP datagrams that are larger than the interface MTU but still less than the send MSS because the TCP stack could pass 16K or 8K bytes of data down the stack to IP. In Host B's case, packets could be fragmented twice, once to get onto the Token Ring LAN and again to get onto the Ethernet LAN.

The procedure flows as follows:

1. Host A sends its MSS value of 16K to Host B.
2. Host B receives the 16K MSS value from Host A.
3. Host B sets its send MSS value to 16K.
4. Host B sends its MSS value of 8K to Host A.
5. Host A receives the 8K MSS value from Host B.
6. Host A sets its send MSS value to 8K.

In order to assist in avoiding IP fragmentation at the endpoints of the TCP connection, the selection of the MSS value is changed to the minimum buffer size and the 40 bytes smaller than the MTU of the outgoing interface. MSS numbers are 40 bytes smaller than MTU numbers because MSS is just the TCP data size, which does not include the 20 byte IP header and the 20 byte TCP header. MSS is based on default header sizes; the sender stack must subtract the appropriate values for the IP header and the TCP header depending on what TCP or IP options are being used.

The way MSS now works is that each host will first compare its outgoing interface MTU with its own buffer and choose the lowest value as the MSS to send. The hosts will then compare the MSS size received against their own interface MTU and again choose the lower of the two values.

Figure 2 illustrates this additional step taken by the sender to avoid fragmentation on the local and remote wires. The MTU of the outgoing interface is taken into account by each host (before the hosts send each other their MSS values) and how this helps avoiding fragmentation.

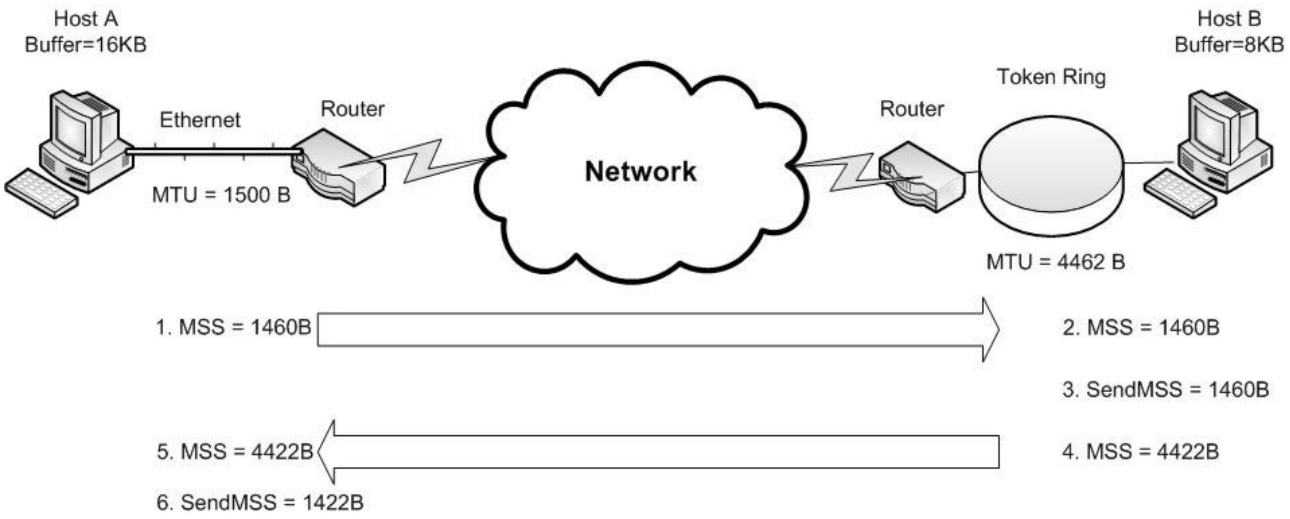

Fig. 2. Avoiding fragmentation using MSS

1. Host A compares its MSS buffer (16K) and its MTU (1500 − 40 = 1460) and uses the lower value as the MSS (1460) to send to Host B.

2. Host B receives Host A's send MSS (1460) and compares it to the value of its outbound interface MTU − 40 (4422).

3. Host B sets the lower value (1460) as the MSS for sending IP packets to Host A.

4. Host B compares its MSS buffer (8K) and it's MTU (4462−40 = 4422) and uses 4422 as the MSS to send to Host A.

5. Host A receives Host B's send MSS (4422) and compares it to the value of its outbound interface MTU −40 (1460).

6. Host A sets the lower value (1460) as the MSS for sending IP packets to Host B.

In this scenario fragmentation does not occur at the endpoints of a TCP connection because both outgoing interface MTU's are taken into account by the hosts. Packets can still become fragmented in the network between Router A and Router B if they encounter a link with a lower MTU than that of either hosts' outbound interface.

There may be some problems if using this method only for avoiding IP fragmentation. The first one is that the MSS option works for TCP protocol only. This means that for other type of traffic, including UDP, RTP over UDP which is common for the IP Telephony applications, and even encapsulated TCP traffic, which means TCP protocol segments which pass some kind of tunnel, including VPN connection etc. are vulnerable to the fragmentation process and we can not rely when using this approach will successfully avoid fragmentation.

Another disadvantage of this approach is that it guarantees fragmentation does not occur at the endpoints of a TCP connection because both outgoing interface MTU's are taken into account by the hosts. But packets can still become fragmented in the network between Router A and Router B if they encounter a link with a lower MTU than that of either hosts' outbound interface.

And at last the TCP MSS Option exchange is performed only in TCP SYN segments, which means only when establishing the connection. The path between two hosts may change without notice after the connection is already established, and fragmentation may occur on the new path. Thus we can expect already established connection to slower the data exchange or even to drop because of fragmentation process.

We must also notice that the TCP level implementation is commonly build into the operating system's design and for the end user is difficult if not impossible to control this process.

However there is an option to setup the receiver's router to manually adjust the TCP MSS value, overriding the sent from the corresponding side value. Knowing the network topology an administrator may manually set the router's MSS value which are returned to the client side and thus to manually prevent the fragmentation occurrence. An example of such process is shown on figure 3.

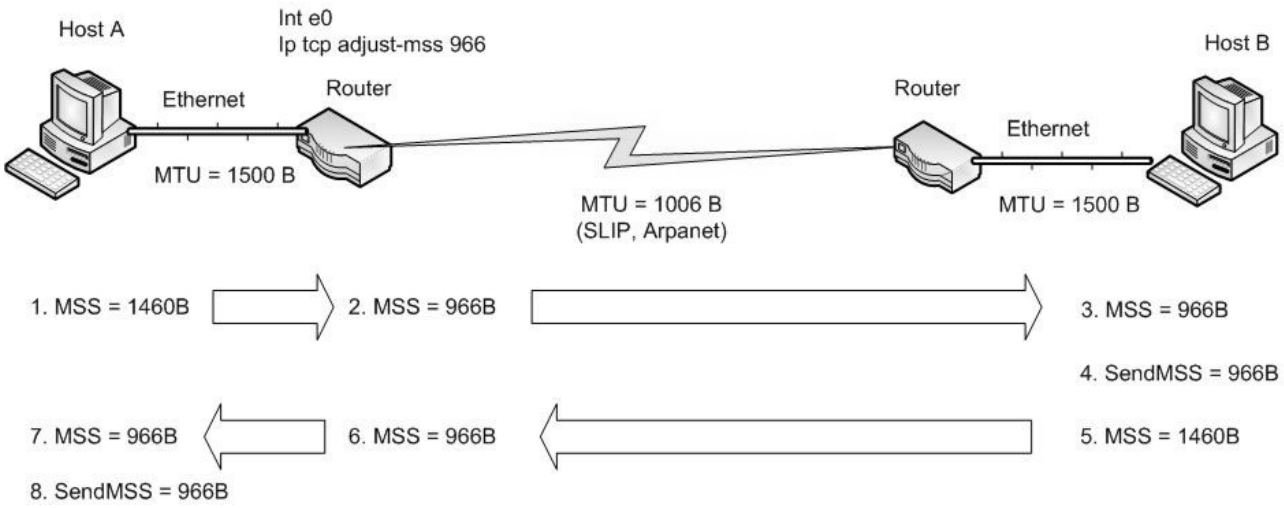

Fig. 3. Router assisted avoiding of fragmentation

1. Host A compares its MSS buffer (16K) and its MTU (1500 − 40 = 1460) and uses the lower value as the MSS (1460) to send to Host B.

2. The router overrides Host A's sent MSS (1460) with administratively assigned value 966 (1006 – 40 = 966).

3. Host B receives Host A's send MSS (966) and compares it to the value of its outbound interface MTU − 40 (1460).

4. Host B sets the lower value (966) as the MSS for sending IP packets to Host A.

5. Host B compares its MSS buffer (8K) and it's MTU (1500 − 40 = 1460) and uses 1460 as the MSS to send to Host A.

6. The router again overrides the sent value with administratively assigned value.

7. Host A receives MSS (966) and compares it to the value of its outbound interface MTU − 40 (1460).

8. Host A sets the lower value (966) as the MSS for sending IP packets to Host B.

In this example is shown how the router can assist for avoiding fragmentation. It is necessary to know the network topology for using this method, but it is a very fast method for avoiding described process for the whole network while configuring a single device – the router.

CONCLUSIONS AND FUTURE WORK

The main benefit using this approach for avoiding IP fragmentation process is that there is no need of existing protocol changes, neither of developing new surrounding protocols. Using pre-built mechanisms is quiet simple as changing a registry key or changing a line in a configuration file.

The present paper aims to describe a possible method for avoiding the process of IP fragmentation and reassembly. Considering the drawbacks above there are some possibilities to use this simple option for certain type of traffic or in combination with some other fragmentation avoiding mechanism.

REFERENCES

[1] IETF, "IP Fragmentation and Reassembly", http://www.ietf.org/rfc/rfc791.txt.

[2] IETF, "The TCP Maximum Segment Size and Related Topics", http://www.ietf.org/rfc/rfc0879.txt

[3] Kent, C., J. Mogul, "Fragmentation Considered Harmful", Digital Western Research Laboratory.

ABOUT THE AUTHORS

Assistant Prof. Delian Genkov, Department of Computer Systems and Technologies, Technical University of Gabrovo, Phone: +359 66 804666, E-mail: dgenkov@tugab.bg.

Assoc.Prof. Raycho Ilarionov, PhD, Department of Computer Systems and Technologies, Technical University of Gabrovo, Phone: +359 66 223209, E-mail: ilar@tugab.bg.