Avoiding the IP Fragmentation Process at the Application Layer of the OSI Reference Model

Delyan Genkov, Raycho Ilarionov

Abstract: *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 some considerations about implementing the IP fragmentation avoiding at the Application Layer.*

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

INTRODUCTION

The fragmentation and reassembly function ensures compatibility of different network architectures, connected in an internetwork when the different networks in a route supports different maximal datagram size. Fragmentation of an Internet datagram is required when it comes from a network that permits a large-sized packet but it should pass through a network that limits the package size in order to reach its destination. Although the IP protocol requires a gateway to fragment a packet if it is too large to be transmitted, it can lead to poor performance or complete communication failure. [4]

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. [4]

The present study aims at brief considering of the possible layers to implement avoiding the fragmentation process, to compare the existing solutions and to propose a method for implementing the avoiding of the IP fragmentation process at the application layer of the OSI reference model. [1]

AVOIDING FRAGMENTATION AT THE NETWORK LAYER

The original recommendations regarding avoiding the fragmentation process were focused at the network layer. They believe that that the right place for fragmentation avoidance is the layer common to all IP communication, it is the IP datagram layer itself (and its partner, the ICMP protocol) or if using the OSI terminology it is the network layer.

There was some proposals in the past, including "Probe MTU", or "Internet Probe Message Protocol" [4], but most of them required one or more changes in the existing protocol specifications or development and implementation in the routers and hosts of a new protocols. Because it is very difficult to change the whole Internet just because avoiding fragmentation, while most applications works well in almost all the cases with fragmentation, most of these proposals were never been realized in practice.

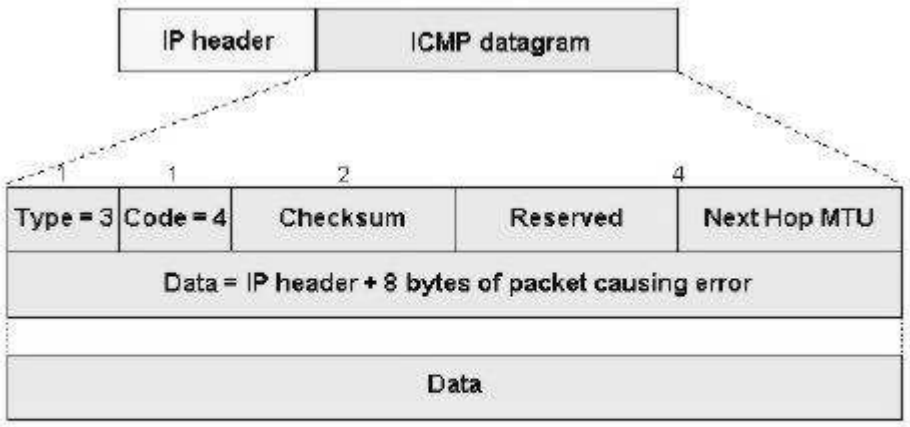

Fig. 1. ICMP Message "Fragmentation Needed and DF Set"

Because the only option to dealt with fragmentation built in the IP protocol is DF (Don't Fragment flag) in the IP header and for the ICMP protocol there is an error message, called ''Destination Unreachable /Fragmentation Needed and DF Set'', the only possible strategy for avoiding fragmentation without protocol changes is the Path MTU Discovery process described in [2], and widely used in the modern operating system's design.

This process presumes always to send IP packets with DF bit set, which means to disable fragmenting these packets. When a router must forward a packet through a path with a smaller MTU but the DF bit is set, the router must discard the packet and to return the sender the ICMP error message shown above. Thus the sender knows that the sent packet size can not travel this route without fragmentation and must smaller the packet's size. This process is done until detecting a size of the packets to be as large as possible while still avoiding fragmentation.

There are three things that can break PMTUD, two of which are uncommon and one of which is common.

- A router can drop a packet and not send an ICMP message. (Uncommon)
- A router can generate and send an ICMP message but the ICMP message gets blocked by a router or firewall between this router and the sender. (Common)
- A router can generate and send an ICMP message, but the sender ignores the message. (Uncommon)

The first and last of the three bullets above are uncommon and are usually the result of an error, but the middle bullet describes a common problem. People that implement ICMP packet filters tend to block all ICMP message types rather than only blocking certain ICMP message types. A packet filter can block all ICMP message types except those that are "unreachable" or "time−exceeded." The success or failure of PMTUD hinges upon ICMP unreachable messages getting through to the sender of a TCP/IP packet.

Many network administrators have decided to filter ICMP at a router or firewall. There are valid (and many invalid) reasons for doing this, however it can cause problems. ICMP is an integral part of the Internet and can not be filtered without due consideration for the effects. In this case, if the ICMP can't fragment errors can not get back to the source host due to a filter; the host will never know that the packets it is sending are too large. This means it will keep trying to send the same large packet and it will keep being dropped--silently dropped from the view of any system on the other side of the filter. While a small handful of systems that implement PMTUD also implement a way to detect such situations, most don't and even for those that do it have a negative impact on performance and the network.

Another reason for missing the ICMP message is that on many routers, a separate IP address in the same subnet is required for each end of a point to point link. This can use address space if there are a large number of such links. Since the actual address of the links doesn't appear to impact much, many people use RFC 1918 private address space for such links. If using such addresses, then ICMP messages (including "can't fragment" errors) will normally be generated with private source address. Since many networks filter incoming traffic from such reserved addresses, the result is the same as if all ICMP were being filtered and can cause the same problems.

If this is happening, typical symptoms include the ability for small packets (e.g. request a very small web page) to get through, but larger ones (e.g. a large web page) will simply hang. This situation can be confusing to the novice administrator because they obviously have some connectivity to the host, but it just stops working for no obvious reason on certain transfers.

Although it is not explicitly specified, the most of the current realizations of the PMTUD algorithm works only for TCP traffic, because they rely on the TCP acknowledgements for retransmission of the missed packets. 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 that the PMTUD process will successfully avoid fragmentation.

The main problem with PMTUD actually is that the IP layer is built in the operating system's development and in the most operating systems there are less or even none parameters to tune regarding IP fragmentation process, we can not effectively dealt with the problems like these above.

AVOIDING FRAGMENTATION AT THE TRANSPORT LAYER

The next internet layer with some possibilities for avoiding IP fragmentation is the transport layer and the TCP protocol option, called MSS [4]. 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.

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.

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.

There are again three problems if using this method for avoiding IP fragmentation.

The first one is that the MSS option works for TCP protocol only. Thus for some other type of traffic, including UDP, RTP over UDP, and even encapsulated TCP traffic are vulnerable to the fragmentation process and we can not rely that we are avoiding 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.

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

AVOIDING FRAGMENTATION AT THE APPLICATION LAYER

The following proposed method for avoiding IP fragmentation at the application layer is based of the Path MTU Discovery algorithm [2]. The main differences from the original method are two. At first there are proposed possibilities to overcome the problems with the original algorithm shown above, at second this algorithm can be built in the application itself, thus made the application independent from the operating system, transport protocol used, etc. That could be very useful for developing platform independent applications, building this approach in the application can make it to behave the same way independent of the operating system used.

In order to detect fragmentation not only in the beginning but in the whole communication process all the packets are send with "Don't Fragment" bit set, which means that if a router must transmit the packet over a link with a smaller MTU it will drop the packet and if not filtered it will send back an ICMP message "Destination Unreachable /Fragmentation Needed and DF Set''. We must never use packet size smaller than 576 bytes, since this is the official smaller MTU in the Internet and every router must not fragment a packet with this size.

In the beginning the application (even if not needed) must send a packet with size equal to the local interface MTU. If the packet is confirmed correct that is the size we will use upon the next detection process.

If the packet is too large to be transmitted without fragmentation, then the router will drop the packet. There are three possibilities for response:

- Router will return ICMP error message, which will contain the "Next-hop MTU" [2], which means the MTU of the link. In this case that is the size of the packet we must use for the next try. Unfortunately this is done by relatively small percentage of the routers.

- Router will not send ICMP message due to restriction, or the error message will be filtered by a firewall. In this case the application must run a recovery procedure. It assume to wait for a timeout where the maximum time to wait must be an adjustable parameter,

because the timeouts for different Internet connection types are variable. Because missing one ICMP message is not a criteria for a miscommunication, the application must resend the packet again, until success, error or reaching maximal number of tries allowed, which must be another configurable parameter. The possibilities here are success – then we use this size of the packet; error message or reached maximum tries – must use smaller packets.

- The most common possibility is the router returns an error message, but the message does not include the "Next-hop MTU" parameter. In this case we must immediately lower the size of the packets sent.

In the last two cases we must lower the packet size, without knowing the actual size. Proposed are three common strategies the application may follow:

- To lower the size by a constant, e.g. size = 0.75 * size. This approach suppose smaller number of iterations to find a size that not permit fragmentation, but usual will not find the exact MTU size, which means it will not use the most effective size of the packets;

- To try a binary search of the maximal packet size, can be sent without fragmentation. This approach will always find the exact MTU size, but sometimes for the cost of many iterations and the discovery process may be slower;

- To have a table with most common sizes used in various networks. This approach combines fast finding (smaller iterations) of a size that not allows fragmentation with finding the exact or too near to it size. The disadvantage of this approach is less flexibility – when using a new technology or a modified one (with different MTU size) the modifying of the tables with the most common MTU sizes is required.

In any of the cases the result is a size the application must use for the packets sent. This size may be unique for a destination, which means for an IP address. The proposal is to store these values in a table for a faster finding if a new session must be done with an already known destination. Thus when sending a packet the application must first look in the table to find known destination and if not fount it must start the discovery procedure.

The entries in the table must be time stamped when added and must expire after time. The exact value of the expiration timer may be another configurable value. When expired, the entry must be removed from the table, which means that for the next packet the application will start the discovery process again. This can help to detect an increased MTU to this destination. The estimated timeout interval may be in order of ten minutes. Any time an ICMP error message "Destination Unreachable /Fragmentation Needed and DF Set'' is received the application immediately lowers the sending datagram size by the method chosen.

In any case when a retransmission is needed due to an ICMP error message or due to an unacknowledged segment for the congestion avoidance reason the application must do a "Slow Start", which means to retransmit one segment at a time but not the whole TCP Window size, until the acknowledgements received.

In order to improve the detection process instead of purging the information for a path and starting the whole discovery process again, the application may raise the size of the packets sending for that destination to the next higher size from the table. Thus the discovery process will be done from two closer values and will be done faster. The timeout value also may vary for a better discovery. It may lower when the size increased, because in this case is more likely to raise fragmentation, and may get longer when the size is decreasing.

The proposed design is not without some drawbacks; most of them will be discovered and possibly avoided in the future works related to this study. Now we can define two possible attacks. The first possible situation is when for some kind of reason someone returns a MTU size quiet smaller size than reality, that means smaller than the actual size, but larger than the official smaller datagram in the Internet – 576 bytes. In this case the actual data flow will pass through the route without causing any errors, but the

actual throughput will be greatly reduced due to inefficient use of the internet resources. It is possible to fall in the contrary situation when someone returns a MTU quiet greater than reality. In this situation temporary blockage of the data flow is expected, due to many packets are dropped during transmission.

CONCLUSIONS AND FUTURE WORK

The present paper aims to describe the current possible methods for avoiding the process of IP fragmentation and reassembly, do define their strengths and drawbacks and finally to propose a new method for avoiding this process. Avoiding fragmentation at the application layer will allow the application designers to design platform independent applications which performed well under any kind of operating system, transport protocol ant internetwork architectures.

There are many future works to do before these considerations are taken to reality, first of all are to build an application having such logic and research the actual behaviour under different situations.

REFERENCES

[1] Cisco Systems, "The OSI Reference Model", http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/introint.htm

[2] IETF, "Path MTU Discovery", http://www.ietf.org/rfc/rfc1191.txt

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

[4] Kent, C., J. Mogul, "Fragmentation Considered Harmful", Digital Western Research Laboratory, 1987

ABOUT THE AUTHORS

Assistant Prof. Delyan 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.