Dynamic MTU : Technique to reduce packet drops in IPv6 network resulted due to smaller path mtu size

Ishfaq Hussain†, Dr. Janibul Bashir∗,†

Abstract – With an increase in the number of internet users and the need to secure internet traffic, the unreliable IPv4 protocol has been replaced by a more secure protocol, called IPv6 for Internet system. The IPv6 protocol does not allow intermediate routers to fragment the on-going IPv6 packet. Moreover, due to IP tunneling, some extra headers are added to the IPv6 packet, exceeding the packet size higher than the maximum transmission unit (MTU), resulting in increase in packet drops. One probable solution is to find the MTU of every link in advance using the Internet Control Message Protocol (ICMP) packets and accordingly fragment the packets at the source itself. However, most of the intermediate routers and the network firewalls do not allow ICMP packets to traverse through their network, resulting in network black holes, where we cannot know the MTU of some links in advance. This method tries to handle the packet drops in IPv6 network by proposing a DMTU scheme where we dynamically adjust the MTU of each link depending upon the original size of the IPv6 packet, thereby reducing the number of packet drops by a significant amount. Using mathematical and graphical analysis, our scheme proves to be much more efficient than the state-of-the-art PMTUD scheme. In this paper the method, mathematical and graphical representations are focusing solely in IPv6 Internet communication.

Keywords – Packet Drop, MTU, IPv6 Protocol, Dynamic MTU, PMTUD algorithm, path MTU

1 Introduction

Due to rapid increase in the number of devices connected to the internet, the current addressing scheme Internet Protocol Version 4 (IPv4) won’t be able to provide addresses to all the devices [1]. It is because IPv4 uses 32-bit addresses, thus, less than 5 billion devices can be addressed using this scheme. And this limit has already been crossed by the number of internet enabled devices. Thus, there is a need of a new addressing scheme. Fortunately, Internet Corporation for Assigned Names and Numbers (ICANN) has already announced a replacement to the IPv4 protocol, called IPv6 [2]. It uses 128-bit addressing scheme, which can address upto

∗Corresponding author. Email: janibbashir@nitsri.ac.in
†Department of Information Technology, National Institute of Technology, Srinagar, India
\(9^{128} \approx 340\) undecillion end devices [2]. Besides, addressing trillions of devices, IPv6 provides a framework for a secure communication over an insecure network. It does this by using an add-on security protocol, called IPSec [3]. For the proper functioning of IPsec, the IPv6 does not allow intermediate routers to fragment the on-going IPv6 packets. This security function is missing in the IPv4 protocol making it vulnerable to certain attacks. These security issue and addressing problem in IPv4 makes the new IP protocol v6 deployment very essential to keep on the smooth working of the Internet system.

Current migration from IPv4 to IPv6 is done through the transition, using principle of tunneling and translation techniques [4] which provides a means to carry IPv6 packets over unmodified IPv4 routing infrastructures and only a very small percentage of the network holds native IPv6. The process of tunnel came at a cost of increased packet size which pushes packet size greater than IPv6 minimum link maximum transmission unit (MTU) [5]. Moreover, a unique feature of NO fragmentation of packet with minimum link MTU of 1280 octets in IPv6 packet which further facilitates the intermediate nodes to drop the packet, which are having low path MTU in their outgoing links.

There are many factors of packet drop by intermediate nodes namely network congestion, Bit Error Rate, TCP MSS size, low Bandwidth, collision. The Internet Control Message Protocol version 6 (ICMPv6) is designed to inform the source nodes of such causes of packet drop by effecting nodes. Based on ICMP messages source node follows different mechanisms in sending the new regenerated packets. Namely, Congestion-control, Hamming-code, FEC, ALOHA and CSMA/CD [6] which addresses to the specific problem of packet drop in the link and many related new researches are proposed to improve these strategies in references [7, 8, 9]. Similarly, the packet drop due to path MTU also make use of ICMP error packet, the intermediate effecting node sends ICMP PTB message to source node to re-transmit the packet. On bases of ICMP PTB message different solution can be used by source node.

One possible solution is to find the MTU of every possible link, using the ICMP packets, in advance and accordingly fragment the packets at the source itself [2]. However, most of the intermediate routers and the network firewalls do not allow ICMP packets to traverse through their network, resulting in network black holes, where we cannot know the MTU of some links in advance. This scheme reduces the frequency of packet drops, but due to the presence of black holes, the number of packet drops is still higher than the tolerable limit.

The other solution is the Path MTU Discovery (PMTUD) protocol [10]. To avoid the IP fragmentation [11], PMTUD determines the relevant MTU between the two IP hosts and accordingly fragments the packet at the source itself. However, this scheme relies on the ICMP packets, which has various issues related to the black holes [12] (please see Section 2).

To overcome the issues, we proposed a technique called Dynamic MTU (DMTU) where we dynamically adjust the MTU of each link depending upon the original size of the IP packet. After receiving the packet, we dynamically adjust the MTU of the outgoing links and then accordingly forward the packet. However, in some scenarios we cannot adjust the MTU of a link above a certain value. In those cases we discard the packet and inform the source to further fragment the packet.
Our main contributions:

A We proposed a technique called DMTU protocol to reduce the frequency of packet drops.

B We proposed different versions of the protocol in order to increase the efficiency of the system and thereby reduce the contention in the network.

C We presented the mathematical evaluation of our scheme and analyzed the frequency of packet drops, time delay, throughput, and latency enhancements.

The rest of the treatise is as follows. Section 2 discusses the related work and motivations. In Section 3 described the working of our proposed DMTU scheme followed by Section 4 discusses analysis on time delay of our proposed method with previous start-of-art method Path MTU Discovery. Graphical analysis and comparison results are presented in Section 5, and the paper concludes in Section 7 with conclusions and future work.

2 Related Work and Motivations

Since to deal with packet drop as a result of Path MTU for IPv6, a proposal was made by \[10\] called "Path MTU Discovery version 6" which is upgraded version of state-of-arts proposed by \[13\] for IPv4. This method works in side by side with fragmentation at source. The main idea of Path MTU Discovery is that the source node initially assumes that the effective MTU, called Path MTU (PMTU), of a path is the (known) MTU of the first hop in the path. If any of the packets sent on that path are too large to be forwarded by some node along the path, that node will discard them and respond back with an ICMPv6 "Packet Too Big" packet.

Upon receipt of such a message, the source node reduces its assumed Path MTU for the path and makes it equivalent to the MTU of the constricting hop as reported in the "Packet too Big" message. Thus, the source fragments the packets to make the size equal to new path MTU. The process ends when the source node’s estimate of the PMTU is less than or equal to the actual PMTU \[10\].

This method is currently deployment for IPv6 network and is up and running, but it suffers from very complicated issues about which we will explain briefly some of major problems in later subsections.

Further work present in \[14\] by Vijay Kumar which is extended in \[15\] by Z. Hu et al. in IETF working group, where the node calculates the Path MTU of the link using the IS-IS routing protocol, by doing so the overhead incurred in the existing path maximum transferable unit discovery mechanism is reduced and the same solution can be extended to other link state routing protocols as well \[14\]. This proposal is still an internet draft and is not approved for a RFC by IETF and hence under development and experimental.

In 2007 \[16\] have came up with a new approach of finding Path MTU released on IETF an internet working group under RFC 4821. This method is named as Packetization Layer Path MTU Discovery (PLPMTUD) which is an extension to RFC 1191 and RFC 1981 which are ICMP based Path MTU Discovery. The PLPMTUD works above the packetization layer in the transport layer (TCP, SCTP, RTP etc.) to dynamically discover the MTU of the path by probing with consecutively larger packets with each successful packet delivery give rise to the packet size equal to the probing packet, without depending on ICMP PTB packets which resolves many of robustness problems which are in the classical PMTUD defined in RFC 1191 and RFC 1981. The Implementation of PLPMTUD is complicated as the probing packet don’t
tell about if it is lost weather its due to the congestion or path MTU issues. Its only deployed in the Linux kernel from version 2.6.17 which is off by default [17].

Currently, in Sep 2020 an updated version of PLPMTUD is released named as Datagram Packetization Layer Path MTU Discovery (DPLPMTUD) in [18], which has same working as in PLPMTUD but works in UDP layer protocol.

Both of the New approaches PLPMTUD and DPLPMTUD need to send probing packets to know the path MTU of the link and is a hit and trail method of knowing the path MTU. Sending probing packets might help in successfully transmission of the original packet with minimum risk of loosing the packet, but with the price of consumption of network resources of sending multiple probing packets. These two methods still dependent on the source node and the usage of probing packets creates same issues like the ICMP-PTB packets which contributes to increased time delay for transmitting a packet. Unlike Classical PMTUD, PLPMTUD and DPLMTUD, our proposed method work on intermediate node level to solve the path mtu with no need of ICMP-PTB or probing packet at source level. The proposed method is designed in such a way that it will work in parallel mode with classical PMTUD and in Standalone mode, without use of Classical PMTUD.

In this study PMTUD is refereed to classical PMTUD throughout the paper which is defined in RFC 1191 and RFC 1981 and don’t implies to the PLPMTUD or DPLPMTUD. Again the PMTUD mentioned in the parallel mode algorithm is referring to the classical PMTUD and motivations for designing the proposed method is also from the failures seen in PMTUD which is currently the working mechanism and didn’t take in-account the PLPMTUD and DPLPMTUD which still needs to resolve and review many issues for implementation.

In the following subsections we have briefly elaborated some of major issue of classical PMTUD which motivated the need of developing a new proposed mechanism which works at intermediate node level in standalone and in parallel with classical PMTUD.

2.1 Path MTU Discovery Vulnerability

The Path MTU Discovery is vulnerable to two types of denial of service attack (DOS). A DOS attacks on a networking structure to restrain or stop the servers from communicating to its clients [19, 20]. Both of these attacks are based on malicious party sending false Packet Too Big (PTB) message by informing smaller or larger Path MTU size than in reality to the source node [10]. In each of these attacks the sender observes sub-optimal performance and temporary blockage and can cause Black hole connection, where TCP hand shaking performs completely but connection hangs during data transfer [10]. A single iteration of PMTUD procedure doesn’t completely assure the transmission of packet to the destination it needs to go for multiple iteration of PMTUD algorithm to successfully sends the packet to destination. Hence the frequent use of the PMTUD by the source node in sending the same packet results in increase in the fragmented packs of small size and consumption of network resources.

2.2 Network Resources Consumption

The nodes using Path MTU Discovery must need to detect decreases in PMTU as sooner as possible. To keep a Path MTU updated the source node continues to elicit Packet Too Big messages then the current estimated path MTU after a time frame of 5 minutes or 10 minutes (as Recommended), this updating can likely be prone to stale and false packets that were floating in the network and those source nodes having multiple paths to the destination, where each carries different Path MTU might result in updating false Path MTU [10]. Beside this, the updating process in which source continue to elicit messages leads to consumes network resources [10].
In *Path MTU Discovery* (PMTUD), packets which are very small in size are forwarded through the network for a long period of time which may result in inefficient usage of the network bandwidth which is the common case in the PMTUD scheme, when path has a node which has much lower MTU then the other nodes [10, 14].

The PMTUD uses ICMP message to know the Path MTU of the link, but the ICMP packet has to travel from problem occurred node to the source node which consumes considerable amount of bandwidth on all the intermediate links between the problem occurred node and the source node [14].

### 2.3 Path MTU Black Holes

In [12] claims that the main cause for the occurrence of PMTUD black holes in the Internet is the filtering of important signalling packets by the intermediate nodes on the path. In the event that these nodes run a firewall, they could potentially be configured to disallow all or certain types of Internet Control Message Protocol (ICMP) packets to pass through them. The effect of this filtering results in the inoperable of *Path MTU Discovery* (PMTUD) and this failure of PMTUD will lead to black holes [21].

Most of the routers give low priority to the ICMP messages carrying the information of the dropped packets. In addition, most of the users and ISP providers configure their firewalls to block all ICMP messages [21] because it wastes the network resources and creates congestion in the network [22]. Thus, due to low priority and blocking of ICMP messages, the ICMP messages sent in the PMTUD will be either delayed or blocked from reaching the source node, resulting in either the increased delay or the ICMP black-holes [23, 21].

This will lead to inoperable and connectivity failure. Due to ICMP message unreachable problem, the host will continuously keep on sending packets and will eventually lead to increased congestion in the channel, thereby reducing the overall performance of the system.

### 2.4 Path MTU Discovery Incompatibility with IPv6

The Path MTU Discovery [13], with the base design is made keeping the features and characteristics of the IPv4 protocol without taking IPv6 characteristics in account. Suppose, if a link has Path MTU lesser then minimum Path MTU of IPv6 packet then in that case how the situation can be handled? In such scenario, sending nodes violate the IPv6 constrains by generating packets of size lower than the prescribed minimum Path MTU of IPv6 packet [2] widely in IPv6 network tunnelling, which in further results in wastage of bandwidth usage [24]. Which make Path MTU Discovery far more less effective in handling packet drop due to Path MTU in IPv6 network. Keeping this issue in one hand another factor that slows the PMTUD in IPv6, that the ration of lowest path MTU of IPv4 to the standard MTU of link is much smaller than the ratio of lowest path MTU of IPv6 to the standard MTU of link, i.e.

\[
\frac{\text{LowestPathMTU}(IPv4)}{\text{standard}(MTU)} \ll \frac{\text{LowestPathMTU}(IPv6)}{\text{standard}(MTU)}
\]

Since we haven’t seen any change in Standard MTU value for IPv6 network , after the minimum packet size raised from 576 octets to 1280 octets in IPv6 [2]. In prior IP version even a lower packet size of 576 bytes can be forwarded or transmitted [25] without any fragmentation or packet drop, while that’s not in case of IPv6. Like a case in DNS which don’t accepts packet size greater than 512 bytes [26] and a IPv6 source node is not allowed to create or generate a packet of size lower than 1280 octets which makes DNS incompatibility issue in IPv6 network [27].
This shows that a small range of packet size from 1280 to 1500 octet can only be transmitted to the IPv6 network. While packets greater than 1500 octet get an ICMP error message and packets lower than 1280 octets are restricted [2]. Since, there is no any other strategy for the source node other then fragmentation of the packet after getting ICMP PTB message due to Path MTU and in some cases, it is not allowed to do fragmentation of packets which are not a factor of 1280 octet because it results in generating packet of size lower than 1280 octet [28]. In IPv4 network, if a source node generates a packet in range of 576 to 1500 octets and during transmission the packet encountered a node having smaller MTU then it has the permission to fragment the packet if Don’t fragment bit is not been set by the source node [25]. Thus PMTUD in IPv4 has to handle only packets which have DF bit set which is not in case of IPv6 network where packet can’t be fragmented by intermediate nodes & the source node can’t fragment lower than 1280 octet which further makes the IPv6 packets vulnerable to be dropped by the nodes in the path and make increase in workload of PMTUDv6 then in PMTUDv4. Despite such compatibility issues, PMTUDv6 in IPv6 network operates in a large range of packet size $Y$ such that,

$$Y : U \setminus \{1280, \ldots, 1500\}$$

where, $U \in \mathbb{N}^+$ whereas, in PMTUDv4 in IPv4 network operates in a smaller range of packet size $X$ respective to PMTUDv6 such that,

$$X : U \setminus \{576, \ldots, 1500\}$$

In general which further makes Path MTU Discovery less effective and low efficient technique in IPv6 network.

Since from all of these drawback in previous proposed inventions ([10], [28] and [14]) no new invention or scheme is initiated or proposed in intermediate node level to get through these issues in IPv6 protocol. In this invention we initiate an algorithm in the intermediate node on receiving such large packets then Path MTU on the fly to prevent packet drop and further optimise the load due to re-transmissions, probing, maintenance of Path MTU value and fragmentation’s in source node. In our proposed method we tried to reduce the usage of ICMP message to a far very low or even to null value. In this method, the intermediate nodes process the packet on the fly without violating any network policies and regulations, and hence reaches to maximum throughput and lower latency of network. Thereby rescuing a lots of packets in IPv6 network to be the victim of Path MTU.

### 3 Proposed Mechanism

In this paper we present a Dynamic MTU scheme, in which it dynamically adjust the MTU of the intermediate links based on the size of the received packet. As soon as the intermediate router transmits a packet to the next forwarding interface link, it reverts back the MTU to its original value. In Figure 4 describes the DMTU algorithm in two phases: Pre-DMTU and Post-DMTU phases. The algorithm starts with the Pre-DMTU phase and then moves to Post-DMTU. In the Pre-DMTU phase, the algorithm checks the basic conditions necessary for the DMTU functionality such as whether the MTU of the intermediate links can be changed or not. When the Pre-DMTU signals the success (all conditions are satisfied) then only the algorithm moves to the Post-DMTU phase, otherwise it stops the algorithm and drops the packet and informs the receiver about the same. The main aim behind dividing the algorithm into two phases is to avoid the unnecessary computations in case any necessary condition is violated. Thus, decreasing the overheads on the node.
In Figure 1 illustrates the working of DMTU algorithm inside an intermediate node in general form, that when it receives a packet larger than then next forwarding link MTU, it examine the packet then based on the examination it either passes the packet to next Interface link n/n or drops the packet on failing the examination. If a flow has multiple packets then the algorithm is called recursively using More fragment bit property of packet only, to overcome the load and time in examining.

The DMTU algorithm can be used either in parallel with the PMTUD scheme or it can be used as a standalone scheme. In the former case, we call the scheme as Pre-Parallel DMTU and the later one is called Standalone DMTU. In Pre-Parallel DMTU algorithm, we allow DMTU algorithm to run only for high priority packets and whereas all other packets are handled using the PMTUD scheme. However, in the standalone-DMTU, DMTU completely replace the PMTUD algorithm and takes the charge.

Note that the there is a minor difference between the Pre-Parallel DMTU and Standalone DMTU algorithms. The main motive behind having two schemes is to allow the algorithm to work in the networks where the PMTUD algorithm has already been deployed.

3.1 DMTU Phases

The DMTU algorithm can be used either in parallel with the PMTUD scheme or it can be used as a standalone scheme. In the former case, we call the scheme as Pre-Parallel DMTU and the later one is called Standalone DMTU. In Pre-Parallel DMTU algorithm, we allow DMTU algorithm to run only for high priority packets and whereas all other packets are handled using the PMTUD scheme. However, in the standalone-DMTU, DMTU completely replace the
PMTUD algorithm and takes the charge. Note that the there is a minor difference between the Pre-Parallel DMTU and Standalone DMTU algorithms. The main motive behind having two schemes is to allow the algorithm to work in the networks where the PMTUD algorithm has already been deployed.

3.1.1 Pre-DMTU Phases

Figure 2 illustrates the working of Pre-DMTU Phases. This phases carries one argument (*X) as the pointer to the packet in the buffer memory. This phase triggers on receiving an incoming packet of size greater than the next-interfaces MTU. There are two type of phases, PreStandalone DMTU phase and PreParallel DMTU phase, both of which works differently depending upon the type of strategy we are assuming. Both of the phases carries a pointer to the packet (*X) from the buffer memory to be processed. These phases act as an preliminary examination of the packet to run the PostDMTU phase.

**Pre-Standalone DMTU Phase:** Algorithm 1 shows a pseudo-code of the PreStandalone DMTU phase: It works in parallel with Path MTU Discovery. When a new packet invokes, it compares the size of the packet with the threshold and Maximum MTU limit of the path. If the packet size exceeds these limits then it is not possible to successfully route the packet. Instead we force the router to drop the packet (without informing the source). The source will itself re-transmit the packet by fragmenting it into smaller packets after the timeout. However, if the packet size is lesser than the Maximum-MTU then the algorithm moves to the Post-DMTU phase.

**Input:** A pointer to the packet X at buffer memory such that \(\text{Size}(\ast X) > \text{MTU}_{n/n}^{\text{Curr}}\)

**Output:** Parameters \(\alpha, \beta\) and \(C\) with a pointer \(\ast X\) to PostDMTU

\[
a \leftarrow \text{Size}(\ast X);
\]

\[
\text{if } \text{Ver} = 6 \text{ then}
\]

\[
\quad \text{if } a \leq \text{threshold } \&\& a \leq \text{MTU}_{n/n}^{\text{Max}} \text{ then}
\]

\[
\quad \text{PostDMTU}(\ast X, 0, 0, 0);
\]

\[
\quad \text{else}
\]

\[
\quad \text{TIMEOUT};
\]

\[
\text{end}
\]

\[
\text{else}
\]

\[
\quad \text{TIMEOUT};
\]

\[
\text{end}
\]

**Algorithm 1:** PreStandaloneDMTU(*X)

**Pre-Parallel DMTU Phase:** Algorithm 2 shows the pseudo-code of PreParallel DMTU phase which allows DMTU to work only for the high priority packets in IPv6 and Tunnelled packets with proto-41.

Thus, the job of Pre-DMTU phase in this scheme is to first find the priority of the packets. Based on the packet priority it either directs the PMTUD algorithm to take the control or it moves to the Pre-DMTU next step.

In Pre-DMTU next step, it compares the size of the packet with the threshold value. If the packet size is less than the threshold value then it moves to the \(\text{PostDMTU}(\ast X, C, \alpha, \beta)\) phase by returning parameters \(\ast X, 0, 0, 0\), else it discards the packet and informs the PMTUD scheme to send an ICMP error message "packet too big" to the source.
3.1.2 Post-DMTU Phase

Figure 3 & algorithm 3 illustrates the working of PostDMTU phase, which is the heart of the algorithm. Its main function is to change the state of the MTU of the forwarding port to a higher state according to the incoming packet size up-to a threshold limit and forward the packet through the next-Interface link, after that reverting MTU of the forwarding link to its initial state.

When a packet pass the PreDMTU phase, then it will be the invoking packet of PostDMTU algorithm, the algorithm then follows next steps by storing the initial/original MTU of next-Interface and Identification number of packet in some variables, followed by overriding the MTU of the forwarding-Link according to the packet size at location (\(\ast X\)) in buffer memory. Since, the invoking packet may be the fragment of a flow or a single packet without multiple fragments, therefore using the MF bit from the header of the packet, the algorithm follows different steps.

If there are multiple fragmented packets from a flow than for every fragment we need to do an
Input: A pointer to the packet $X$ at buffer memory such that $\text{Size}(\ast X) > MTU_{\text{Cur}}^{\text{n/n}}$  

Output: Parameters $\alpha$, $\beta$ and $C$ with a pointer $\ast X$ to PostDMTU

$a \leftarrow \text{Size}(\ast X)$;

if $\text{Ver} = 6 \text{ || } \text{PN} = 41$ then
  if $\text{Priority} = \text{High}$ then
    if $a \leq \text{threshold} \text{ && } a \leq MTU_{\text{Max}}^{\text{n/n}}$ then
      $\text{PostDMTU}(\ast X, 0, 0, 0)$;
    else
      $\text{ICMP}$;
    end
  else
    $\text{ICMP}$;
  end
else
  $\text{ICMP}$;
end

Algorithm 2: PreParallelDMTU(*X)

---

overriding of MTU followed by PreDMTU phase examination which could take a large network resource and processing time. That’s why we take care of this multiple fragments of same flow using parameter triplets $(C, \alpha, \beta)$ also figure 4 describes the working of DMTU Phases with Multiple fragments.

a). At More fragment = 0: If the MF bit is not set i.e MF=0, then the packet may be
the initial packet or the final packet of a flow. Both types of packet overrides the next interface MTU according to the packet size and then forwards the packet to next interface after which it resets the MTU to its initial state. But the packet which is last packet of a flow needs to compare its ID with initial packet of the same flow if fails it reset the MTU to initial state.

b). At More fragment = 1: If MF bit is set i.e $MF = 1$, then it forwards the packet to next-interface link. After forwarding the packet in location $^X$ in buffer memory the location is now empty and the next packet in the queue takes the location of the previous forwarded packet. To process multiple packets/fragments we makes a recursive call to itself, on doing so it again repeat the same process for the new arriving packet from the buffer memory of storing ID, MTU of next interface and overriding the next Interface MTU. But initial PostDMTU call have already changed the MTU of the next interface MTU, therefore the current running MTU of the interface is not the original MTU and no need to store it in any variable also its not feasible to override again and again the MTU of the next interface according to the packets of same size, its enough to override the MTU according to the initial invoking packet of the flow. Hence, to settle down all these issues we use a parameter $C$ called counter whose value is in $\{0 \text{ or } 1\}$ in PostDMTU.

If Counter value is not set then it means it is the first call from PreDMTU phase, else it is recursive call by PostDMTU which would happen only if there are multiple packets. Therefore after forwarding the initial packet it returns position of next packet as $^X$ and values of initial MTU and ID of the initial invoking packet with a counter value $C = 1$, identifying it as recursive call by PostDMTU. When a recursive call is made to PostDMTU then the next packet to process may be last fragment or may not. If it is not the last packet then this time the algorithm compares the ID of new arriving packet with initial packet from parameter $\alpha$.

The comparison of ID is done to keep running the postDMTU for the flow related to the initial Packet or if the packet router has multiple queues and uses Bit-by-Bit Round Robin (BBRR) then in that case it jumps from one queue to another then the ID of the new arriving
Input:
Parameters $\alpha$, $\beta$ and $C \in \{0, 1\}$. A Pointer to packet $X$ such that $\text{Size}(\ast X) > MTU_{n/n}$

Output: Forward packet at $\ast X$ to $\text{Int}_{n/n}$

if $C = 0$ then
  $r \leftarrow \text{store } MTU_{n/n}$;
  $w \leftarrow \text{store } \text{ID}(\ast X)$;
  $MTU_{n/n} \leftarrow \text{override } \text{Size}(\ast X)$;
end

if $MF = 1$ then
  if $C = 1$ then
    if $\text{ID}(\ast X) = \alpha$ then
      $\text{PostDMTU}(\ast X, 1, \alpha, \beta)$;
    else
      $MTU_{n/n} \leftarrow \text{reset } \beta$;
    end
  else
    $\text{Int}_{n/n} \leftarrow \text{forward } \ast X$;
    $\text{PostDMTU}(\ast X, 1, w, r)$;
  end
else
  if $C = 1$ then
    if $\text{ID}(\ast X) = \alpha$ then
      $\text{Int}_{n/n} \leftarrow \text{forward } \ast X$;
      $MTU_{n/n} \leftarrow \text{reset } \beta$;
    else
      $MTU_{n/n} \leftarrow \text{reset } \beta$;
    end
  else
    $\text{Int}_{n/n} \leftarrow \text{forward } \ast X$;
    $MTU_{n/n} \leftarrow \text{reset } r$;
  end
end

Algorithm 3: PostDMTU($\ast X, C, \alpha, \beta$)
packet may not be the same as the initial invoking packet and a PreDMTU examination is important before calling PostDMTU. If ID’s are not same then it reverts the MTU of port to its initial state using parameter $\beta$, which carries the original MTU of the port, else if ID’s are same then it forwards the packet to next interface link and then do a recursive call to PostDMTU. This time we didn’t stored the ID and MTU value in any variable and instead returning the variables we return the parameters carrying these values $\alpha$ and $\beta$ to PostDMTU along with counter value set and location of next packet. Every recursive call carries the ID and the original MTU of port in $\alpha$ and $\beta$ and it ends when it encounters MF bit 0, which means this is last fragment of the flow and hence after forwarding this packet it reverts the port MTU to its initial state using parameter $\beta$.

In brief, when the new packet arrives, it compares ID number of incoming packet with the stored ID value, if these are same then it forwards multiple packets until MF (more fragments) bit equals to zero. When MF=0, it means no more fragments of the packet are arriving. After this, the algorithm reverts back the MTU to the original value.

The working of Post-DMTU algorithm is same for the Pre-Parallel DMTU and Pre-Standalone DMTU phase.

4 Measuring the Time Delay

In Internet system time is everything, its the basic unit to measure and compare the efficiency and Quality of a network, network parameters and network equipment. So we preferred to proceed first to measure the effects of proposed methods on time delay by comparing it with previous state-of-the-art PMTUD algorithm. Our analysis of time delay for DMTU algorithm make use of the study we carried-out in [29], which have taken a analysis on time delay in PMTUD algorithm in transmitting a single packet of 1800 bytes in a finite network configuration using PMTUD algorithm. We will take the same case study of sending 1800 bytes of packet in same network configuration but using Dynamic MTU algorithm rather Path MTU Discovery for the comparision results. In Figure 5 is a network configuration similar as taken in Figure 1 in [29] for Path MTU Discovery, with same packet size for transmission of 1800 byte of packet from source to destination over a path of varying mtu size using Dynamic MTU algorithm.

The outputs from this case study will be used later in the paper, to further analyse other network parameters and to find out how better and efficient is the new proposed mechanism then the Path MTU Discovery.
4.1 Effect on Total Time Delay Using DMTU

As the analysis led by [29] we will also begin by sending a packet of size 1800 bytes from source to destination over the same network configuration as shown in Figure 5. The packet will pass through the first node N1 and reaches to node second (i.e N2), since the link mtu is lesser then the incoming packet N2 runs the DMTU algorithm and send the packet through the next interface of node N2 to node N3, where in before case study in [29] the node N2 has dropped the packet and needs to send the packet again. Now the node N3 also runs the DMTU algorithm due to exceeding the packet size then its forwarding link mtu and transmits the same packet to next interface of node N3. Then the packet passes from node N4, and node N5 runs the DMTU algorithm and transmits the packet to destination. Since, on sending the packet from the source to destination the DMTU algorithm is triggered 3 times in the nodes where the next interface MTU is lesser the incoming packet size. Therefore, the time for running the algorithm by router is $T_O$ and for three nodes it will be $3T_O$ time which would be added to the total transmission time.

In DMTU case, the first transmission is the first successful transmission without the need of extra fragmentation of packet, while the similar case study in [29] the 4th transmission is the first successful transmission with packet size decreased by 400 bytes. This led to an increase of End-to-End-Delay (E2ED) between nodes $D_T D$ in DMTUD algorithm by a quantity of $D-P_T D$ from the E2ED of successful transmission using PMTUD algorithm in [29]. Therefore, the E2ED of successful transmission (i.e $D_T D$) in using DMTUD algorithm in terms of E2ED of successful transmission (i.e $P_T D$) using PMTUD algorithm in [29] is:

$$D_T D = (P_T D + D-P_T D)$$

The second entity of the Equation 1 i.e $D-P_T D$ is E2ED between nodes for the resultant packet size from subtracting packet size of successful transmission in PMTUD from DMTU i.e packet size in DMTU is taken as D and Packet size in PMTUD is taken as P, therefore the resultant packet size is superscript as $(D-P)$ and its E2ED is taken as $D-P_T D$, which is nearly equals to the difference in time delay of $D_T D$ and $P_T D$.

In play $D_T D \geq P_T D$, therefore the $(D_T D - P_T D)$ is positive and is $< \epsilon$.

i.e

$$D-P_T D = |D_T D - P_T D| < \epsilon$$

$$\Rightarrow \quad D-P_T D < \epsilon$$

The range of $\epsilon$ is not definite, throughout the paper we assume that it is a very small value in fractions that when $\pm \epsilon$ to any natural number $N$ results in minor change after decimal values and the product with any natural number $N$ results value approaches to approximately 0.

i.e

$$x \pm \epsilon \simeq x$$

$$x \epsilon \simeq 0$$

$$\forall \ x \in N$$

The total time delay $D_T D$ of the successful transmission depends upon the overhead time of the node $T_O$ and the E2ED $D_T D$ and no fragmentation time as no fragmentation is carried-out
using DMTU. Therefore, the total time delay in DMTU algorithm would be:

\[
\text{Total time delay } (^D T) = (n + 1)^D T_D + 3 T_O
\]  

(4)

Since, there are only 5 nodes then the Equation 4 becomes:

\[
^D T = 6(^D T_D) + 3 T_O
\]  

(5)

From Equation 1 the total time delay be:

\[
^D T = 6(^P T_D + ^D-P T_D) + 3 T_O
\]

\[
^D T = 6(^P T_D) + 6(^D-P T_D) + 3 T_O
\]

(6)

where, \(|^D-P T_D| < \epsilon|

Since, the total time wastage \(T_W\) will be

\[
^D T_W = 6(^D-P T_D) + 3 T_O
\]

(7)

Therefore, total time delay can be represented as:

\[
^D T = 6(^P T_D) + ^D T_W
\]

\[
^D T = 6(^P T_D) + 6(^D-P T_D) + 3 T_O
\]

(8)

Since the study given by [29] for analysis of PMTUD algorithm the corresponding time delay in PMTUD for the same network configuration is given by :

\[
^P T = (6(^P T_D) + 6(^D-P T_D) + 3 T_O)
\]

(9)

Subtracting the Equation 9 from 8 we get:

\[
^D T - ^P T = (6(^P T_D) + 6(^D-P T_D) + 3 T_O) - (6(^P T_D) + 10(^P T_D + ^P T_D') + 3 T_F)
\]

\[
^D T - ^P T = (6(^D-P T_D) - 10(^P T_D)) + (3 T_O - (10 T_D + 3 T_F))
\]

(10)

Since,

\[6(^D-P T_D) - 10(^P T_D)] < 0[^D-P T_D < ^P T_D]

Also,

\[3 T_O - 10 T_D' + 3 T_F] < 0 \quad ( : [T_D' + T_F] > T_O)

Since, the outcome of the two negative value will be a negative value which implies that :

\[^D T_W < ^P T_W\]

Hence, the total time delay in DMTU is less than the total time delay in PMTUD in a same network configuration which implies that the use of DMTU decreases the time delay. From the above calculations we can show that total time for \(n\) nodes dropping at a nodes can be given by:

\[
\text{Total Time} = [n + 1][^P T_D + ^D-P T_D] + a T_O
\]

where, \(|^D-P T_D| < \epsilon|
4.2 Analysis of Total Time Delay for $n$ - Nodes

The calculations provided in Section 4 are limited to 3 nodes. However, the total time taken by the DMTU and PMTUD for $n$ nodes between source to destination, considering packet drop at ‘a’ number of nodes at position $n_i$th is given as:

For PMTUD as from findings of [29]:

\[ P_T = P_T D(n + 1) + (P_T D + P_T') \sum_{i=1}^{a}[n_i] + aT_F \] (13)

For DMTU:

\[ D_T = P_T D(n + 1) + D-P_T D(n + 1) + aT_O \] (14)

Subtracting Equation 13 from 14 we have:

\[ D_T - P_T = (D-P_T D(n + 1) + aT_O) - \left( (P_T D + P_T') \sum_{i=1}^{a}[n_i] + aT_F \right) \]

\[ = \left( D-P_T D(n + 1) - (P_T D + P_T') \sum_{i=1}^{a}[n_i] \right) + (aT_O - aT_F) \]

\[ D_T - P_T = D-P_T D(n + 1) - (P_T D + P_T') \sum_{i=1}^{a}[n_i] + a(T_O - T_F) \] (15)

Now the term $\sum_{i=1}^{a}[n_i]$ creates two scenarios, consecutive and non-consecutive case scenarios. In consecutive case scenario when packets drop in Consecutively i.e $n_i - n_{i-1} = n_{i+1} - n_i$, where in non-consecutive the packet drop is random and is non-consecutive that is $n_i - n_{i-1} \neq n_{i+1} - n_i$.

4.2.1 Non-Consecutive Case Scenario

The non-Consecutive case scenario arises when the nodes drop packet randomly i.e $n_i - n_{i-1} \neq n_{i+1} - n_i$, then the term $\sum_{i=1}^{a}[n_i] = (n_1 + n_2 + n_3 + n_4 + \ldots + n_a)$. In Figure 6 is the representation of the MTU’s of the nodes in random fashion in a network path in non-consecutive case.

Since, $D-P_T D << P_T D + P_T'$ and the value of $D-P_T D < \epsilon$.

Therefore, the effect of $(n + 1)$ on $|D-P_T D|$ will be:

\[ (n + 1)^{D-P_T D} < \epsilon \]

\[ \Rightarrow \left( (n + 1)^{D-P_T D} \right) < \left( (P_T D + P_T') \sum_{i=1}^{a}[n_i] \right) \] (17)

Hence,

\[ \left( D-P_T D(n + 1) - (P_T D + P_T') \sum_{i=1}^{a}[n_i] \right) < 0 \] (18)

Also,

\[ T_O < T_F \]

\[ \Rightarrow T_O - T_F < 0 \] (19)
Since from Equations 17 and 19 the result of sum of two negative value is a negative therefore:

\[ D_T - P_T < 0 \]
\[ \Rightarrow D_T < P_T, \quad \forall n \in Z^+ \]  

(20)

Hence the total time in the DMTU \((D_T)\) is lesser than total time in PMTUD \((P_T)\) in random case scenario.

4.2.2 Consecutive Case Scenario

In consecutive case scenario when nodes drop packet consecutively \( i.e \ n_i - n_{i-1} = n_{i+1} - n_i \) then the term \( \sum_{\alpha=1}^{a} [n_i] \) = \( S_a = \frac{a(a+1)}{2} \). In Figure 7 show the nodes with MTU in decreasing fashion which give rise to the consecutive packet drop by nodes.

Then the Equation 15 will become

\[ D_T - P_T = D_T T_D(n + 1) - (P_T T_D + P_T' S_a) + a(T_O + T_F) \]  

(21)

From Equation 21 the term:

\[ (D_T T_D(n + 1) - (P_T T_D + P_T' S_a)) < 0, \]
iff,

\[ n + 1 < \frac{a(a + 1)}{2} \]

\[ \Rightarrow \quad 2(n + 1) < a^2 + a \]

\[ \Rightarrow \quad a^2 + a - 2n - 2 < 0 \quad (22) \]

Since the Equation 22 becomes quadratic equation with roots \( \alpha > -\frac{1+(9+8n)^{\frac{1}{2}}}{2} \) and \( \beta > -\frac{1-(9+8n)^{\frac{1}{2}}}{2} \). From the two roots, the root \( \beta > -\frac{1-(9+8n)^{\frac{1}{2}}}{2} < 0 \) which implies that ‘a’ has a negative value, which is not possible as \( a \subset n, n \in Z^+ \). Therefore the root of quadratic equation is \( \alpha > -\frac{1+(9+8n)^{\frac{1}{2}}}{2} > 0 \). The root value has a fractional value and nodes can’t be in fraction, therefore we take the floor value of root \( a \) i.e

\[ \text{Floor}(a) = \lfloor a \rfloor \]

Since the success probability of \( n + 1 < \frac{a(a+1)}{2} \) at \( a > \left[ -\frac{1+(9+8n)^{\frac{1}{2}}}{2} \right] \) is given by probability function:

\[ P(n + 1 > \sum_{i=1}^{a} [n_i]) = 1 - \frac{a - 1}{n} \]

At \( a = \frac{-1+(9+8n)^{\frac{1}{2}}}{2} \),

\[ P(n + 1 > \sum_{i=1}^{a} [n_i]) \leq 1 - \frac{-1+(9+8n)^{\frac{1}{2}} - 1}{n} \]

In Figure 8 shows the success probability by varying the number of nodes, which shows at highest success probability of \( \log_e(2.5173) \) at nodes 300 and lowest success probability of \( \log_e(1.39) \) at node 0, over the number of nodes the term \( \frac{a(a+1)}{2} > (n+1) \) at \( a > \left[ -\frac{1+(9+8n)^{\frac{1}{2}}}{2} \right] > 0, \forall a \subset n \) and \( n \in Z^+ \), which implies that:

\[ D^P T_D(n+1) < (P^T_D + P^T_D')\frac{a(a+1)}{2} \]

Therefore,

\[ D^P T_D(n+1) - (P^T_D + P^T_D')\frac{a(a+1)}{2} < 0. \]

Also, from Equation 19 the term:

\[ T_O - T_F < 0 \]

\[ \Rightarrow \quad D^P T < P^T \quad (23) \]

In Table 1 the data is collected from the success probability graph over number of nodes when \( n \geq n_i > \alpha \), where \( \alpha > \left[ -\frac{1+(9+8n)^{\frac{1}{2}}}{2} \right] \in (n_i) \) at \( \sum_{i=1}^{a} [n_i] \) which shows success probability increases rapidly from 0.3333 to 0.99 as number of nodes increases for \( (n+1) < \frac{a(a+1)}{2} \)
Figure 8 – Success probability graph over number of nodes.

Table 1 – Data from probability graph over nodes.

| Number of Nodes | Success Probability | In Terms of $\log_e(s)$ |
|-----------------|---------------------|-------------------------|
| 0               | 0.3332              | $\log_e(1.3954)$       |
| 1               | 0.4384              | $\log_e(1.5502)$       |
| 5               | 0.6000              | $\log_e(1.8221)$       |
| 10              | 0.6783              | $\log_e(1.9705)$       |
| 20              | 0.7500              | $\log_e(2.1170)$       |
| 30              | 0.7870              | $\log_e(2.1967)$       |
| 40              | 0.8108              | $\log_e(2.2497)$       |
| 50              | 0.8278              | $\log_e(2.2882)$       |
| 100             | 0.8728              | $\log_e(2.3936)$       |
| 200             | 0.9072              | $\log_e(2.4773)$       |
| 300             | 0.9232              | $\log_e(2.5173)$       |

Hence the total time in the DMTU ($^D T$) is lesser than total time in PMTUD ($^P T$) in consecutive case scenario with success probability of $\log_e(2.718)$ for n-nodes.

5 Graphical Analyses and Comparison Results

To find out how much effective and robust is the new proposed method compared to the previous stat-of-art algorithm, we carried out a graphical representation of the network parameters namely Time Delay, Latency and Throughput of both the methods by deriving the related equation of parameters and then compared them. To calculate and compare these parameters we need to find out how the general equations of total time delay of both the algorithms behaves, under some fixed pre-defined values and varying percentage of packet drop. For that we fix the values for the terms of both the general equations of total time delay to be the average/highest limit as seen in real network analyses software’s, so that the results will be accurate.

In [29] shows the complete measurements of the time-delay due to Path MTU discovery in IPv6 network, and we use those calculations for the comparison results. Since the parameters which we are fixing pre-defined value are given by; $^PT_D = 10^{-1}$, $^D-P T_D = 10^{-2}$, $^T_O = 10^{-3}$, $^P T_D = ^P$
Figure 9 – Comparisons of time delay in DMTU and PMTUD.

\[ T_D = 15 \cdot 10^{-2} \]

By keeping these parameters constant and taking iterations of ‘a’ which is the nodes which dropped the packet i.e \( a = (10\%n, 20\%n, 30\%n, 40\%n, 50\%n) \)

We will now be able to get a better graphical representation of these network parameters using these fixed pre-defined value and by varying percentage of packet drop.

5.1 Graphical Analysis of Total Time Delay

In Equation 14 we have Total time delay for DMTU as:

\[ DT = n \left( PTD + D - PTD \right) + \left( PTD + D - PTD + aTO \right) \] (24)

The Equation 24 is in form of \( y = ax + b \), where \( a = (PTD + D - PTD) \) and \( b = (PTD + D - PTD + aTO) \) Taking \( n \) towards x-axis and \( DT \) towards y-axis which gives us a straight line, origin at intercept \( (PTD + D - PTD + aTO) \) of y-axis with (+) slope of \( (PTD + D - PTD) \).

From Equation 13 we have Total time delay for PMTUD as:

\[ PT = PTD(n + 1) + \left( PTD + PTD' \right) \sum_{i=1}^{a} n_i + aTF \] (25)

which also the form of \( y = ax + b \), with \( a = (PTD) \) and Intercept of \( b = PTD + \sum_{i=1}^{a} n_i (PTD + PTD') + aTF \). Where the value \( TF \approx TO \), therefore we take the value of \( TF = TO \) as we now don’t have the overhead time calculated but our assumption is that \( TO < TF \) and for the sake of calculation, we treat it to be equal to \( TF \). Now both the equations follows the linear equation where number of nodes (\( n \)) are along x-axis and Total time delay \( PT \) along y-axis. By subtracting \( PT \) from \( DT \) will also produce a linear equation as:

\[ DT - PT = n \left( D - PTD \right) + \left( \left( D - PTD \right) - \left( PTD + PTD' \right) \left( \sum_{i=1}^{a} n_i \right) \right) \] (26)

Which has a intercept of \( \left( \left( D - PTD \right) - \left( PTD + PTD' \right) \left( \sum_{i=1}^{a} n_i \right) \right) \) and \( a = n \left( D - PTD \right) \).

Now by applying the fixed pre-defined values in both the Equations 24, 25 & 26 by varying percentage of packet drop by nodes we get a spectrum of lines moving upwards, downwards and straight as shown in Figure 9a & Figure 9b.
The Figure 9a is derived at minimum value of \( \sum_{i=1}^{a} [n_i] \), which means the packet drop starts from first node and consecutively till value of \('a'\) which is the last position of the node dropping the packet. In this analysis we see at minimum value of \( \sum_{i=1}^{a} [n_i] \) the time delay for the DMTU algorithm is lower than the PMTUD algorithm. The green line passing closer to the x-axis represents total time delay in DMTU algorithm, while the red lines going in upper directions shows the percentage of nodes dropping packets in different iterations in PMTUD algorithm and blue-line going downward direction along x-axis represents the amount of time delay saved by DMTU algorithm.

The Figure 9b is drawn at maximum value of \( \sum_{i=1}^{a} [n_i] \), i.e. at \( a = n \), which means the packet drops consecutively till the nth node in the path and starts from \( x-a \) value, which represents the highest value of \( \sum_{i=1}^{a} [n_i] \) for nodes that drop packet. This analysis shows that the DMTU algorithm is effective for routes passing large number of nodes with greater probability of packet loss than using PMTUD.

5.2 Graphical Analysis of Throughput Enhancement

The DMTU algorithm increases the throughput with respect to the throughput without DMTU algorithm. And we will show the enhancement in the throughput mathematically.

Mathematically:

\[
\text{Throughput} = \frac{\text{Data Transmitted}}{\text{Unit of Time}}
\]

Let \( P_T R \) be throughput using PMTUD algorithm. Since, Unit of Time = Total Time = \( P_T \) and Data Transmitted = Packet Size

\[
\Rightarrow P_T R = \frac{\text{Packet Size}}{(P_T)}
\]

Let Packet size = \( P \),

\[
\Rightarrow P_T R = \frac{P}{(P_T)} \tag{27}
\]

Let \( D_T R \) be throughput with DMTU,

\[
\Rightarrow D_T R = \frac{\text{Packet Size}}{(D_T)}
\]

Since, packet size = \( P \),

\[
\Rightarrow D_T R = \frac{P}{(D_T)} \tag{28}
\]

In the Equations 28 and 27 of DMTU and PMTUD throughput respectively we derive a graph by fixing a pre-defined value to all constant terms other than nodes. In other words, the number of nodes varies in the route but keep other parameters constant. The value of constant terms will be highest/average limit of expected network simulator results or real time results. We carry this under different packet sizes so we take the iterations of packet size to see the rate of effect of increasing the packet size on the throughput by PMTUD and DMTU algorithms.

Since, on putting values of \( D_T \) and \( P_T \) on Equation 28 and 27 respectively, we have:

\[
D_T R = \left( \frac{P}{n (P_T D + D-P D_D) + (P T_D + D-P D_T D + a T_D)} \right) \tag{29}
\]

\[
P_T R = \left( \frac{P}{(P T_D (n + 1)) + (P T_D + P T_D) (\sum_{i=1}^{a} n_i) + a (T_F)} \right) \tag{30}
\]
The pre-defined constant values are \( P_{T_D} = 10^{-1}, \quad P_{T_D}' = P_{T_D} - (15 \cdot 10^{-3}), \quad D_{T_D} = 10^{-2}, \quad T_O = 10^{-3}, \quad T_F = 10^{-4} \) and let the packet size be constant i.e. \( P = (1300) \). Since, here we take varying value of percentage of nodes dropping the packet to get a clear view of how throughput acts with nodes drop packets.

In Figure 10a shows the throughput between PMTUD and DMTU at single packet at minimum value of \( \sum_{i=1}^{a} [n_i] \). For a single packet at different instant of percentage of nodes dropping packet the PMTUD throughput is lower than DMTU while the DMTU throughput is smooth declining line with higher intercept from x-axis then PMTUD throughput. In other words the DMTU throughput is improved over the PMTUD throughput.

The graph in the Figure 10b shows the throughput between PMTUD and DMTU at single packet at maximum value of \( \sum_{i=1}^{a} [n_i] \). Which shows PMTUD throughput is getting lower for each increase in percentage of node drop packet than DMTU throughput. Since, this graphical observation identifies that the throughput in DMTU algorithm is greater than in PMTUD algorithm for a given number of nodes ‘n’. Again From Equation 20 and 23 we have,

\[
D_T << P_T
\]

\[
\Rightarrow D_{TR} > P_{TR}
\]

\[
5.3 \text{ Graphical Analysis of Latency Enhancement}
\]

Mathematically:

\[
Latency \propto \frac{1}{\text{Throughput}}
\]

The latency is the inverse of throughput, therefore the throughput increases the latency decreases. The latency for the PMTUD algorithm is:

\[
L_P = K \frac{1}{P_{TR}} = K \frac{P_T}{P}
\]

Where \( K \) is constant of proportionality. Now the latency for the DMTU algorithm will be:

\[
L_D = K \frac{1}{D_{TR}} = K \frac{D_T}{P}
\]
let value of \( K = 1 \) for accuracy and putting valued of \( P_T \) and \( D_T \) in Equations 32 and 33 respectively which gives:

\[
L_P = \left( \frac{P_T D_T (n + 1) + (P_T D_T + P_T D_T') (\sum_{i=1}^{n} n_i) + a (T_F)}{P} \right) \quad (34)
\]

\[
L_D = \left( \frac{n (P_T D_T + D_T T_D) + (P_T D_T + D_T T_D + aT_O)}{P} \right) \quad (35)
\]

we have taken the same pre-defined value for the above Equations 34 & 35 i.e. \( P_T D_T = 10^{-1} \), \( P_T D_T' = \frac{P_T D_T - (15 \cdot 10^{-3})}{10^{-2}} \), \( D_T T_D = 10^{-2} \), \( T_O = 10^{-3} \), \( T_F = 10^{-4} \). Like in analyses of throughput, hear we also taken a set of same order of packet size \( P = (1300) \) with varying percentage of nodes dropping the packet.

In Figure 11a shows the latency for the PMTUD algorithm and DMTU algorithm at minimum value of \( \sum_{i=1}^{n} [n_i] \). The latency for PMTUD algorithm is increases with increase in percentage of nodes dropping packet while the latency for the DMTU algorithm is lower than the lowest Latency of PMTUD algorithm with a large Intercept of Y-axis.

Again in Figure 11b shows the latency for the PMTUD algorithm and DMTU algorithm at maximum value of \( \sum_{i=1}^{n} [n_i] \). In maximum value of \( \sum_{i=1}^{n} [n_i] \) the Latency difference between the PMTUD algorithm and DMTU algorithm is larger than the minimum value of \( \sum_{i=1}^{n} [n_i] \).

Again by dividing Equation 32 with 33 we get:

\[
\frac{L_P}{L_D} = \frac{P_T}{D_T}
\]

Since, from Equation 20 and 23 we have:

\[
D_T << P_T
\]

\[
\Rightarrow \quad \frac{L_P}{L_D} > 1
\]

\[
\Rightarrow \quad L_P > L_D
\]

_Hence the Latency with DMTU is Lower than Latency with PMTUD algorithm._

### 5.4 Routing Effectiveness

**Lemma 1.** The ratio of Packet Delivery rate of PMTUD to Packet Delivery rate of DMTU is less than 1.
\[
\frac{PDR_P}{PDR_D} < 1
\]  
(37)

Proof. The number of packets delivered to the number of packets sent to reach to destination by source in same medium is given by Packet delivery rate (PDR) value:

\[i.e. \quad PDR = \frac{\text{Packet Received}}{\text{Packet Send}}\]

such that, \(0 \leq (PDR) \leq 1\)

If the PDR is near to 0, then it shows a less routing effectiveness of protocol and if PDR is near to 1, then it shows higher routing effectiveness of protocol. We now calculate the PDR value for PMTUD and DMTU then compare them to see which has better routing effectiveness. Let’s denote \(PDR_P\) for the PMTUD and \(PDR_D\) for DMTU. Let \(x\) be number of packets to be send by source to destination with intermediate nodes \(n\) and links \((n+1)\) both for DMTU and PMTUD. Let the packets drop due to network factors other than MTU be ‘\(C\)’ and the packets dropped due to MTU-size would be ‘\(l\)’, then the Packets delivered will be \(x-(C+l)\)

\[
PDR_P = \frac{x-(C+l)}{x}
\]

(38)

In DMTU the Packets drop due to factor of MTU or ‘Packet too big’ will be rescued. Therefore, \(l = 0\), while the packets drop due to other factors in network will be same as for PMTUD i.e. ‘\(C\)’.

\[
PDR_D = \frac{x-C}{x}
\]

(39)

Dividing Equation 38 by 39:

\[
\frac{PDR_P}{PDR_D} = \frac{x-(C+l)}{x-C}
\]

(40)

In equation 40 the numerator is less than denominator by value of \(l\), therefore its ratio is less than 1.

\[i.e \quad \frac{PDR_P}{PDR_D} < 1\]

(41)

\[\Rightarrow \quad PDR_D > PDR_P\]

(42)

Hence, which shows that DMTU protocol has higher routing effectiveness than PMTUD. \(\square\)

6 Issues of Implementation of Proposed Mechanism

6.1 Vendor restrictions

The implementation of DMTU can be done by reprogramming the router’s firmware with certain limits to change the MTU depending on the current packet being handled. These limits are
kept to avoid the vulnerability that can be used maliciously. These limits were described in the mechanism. Its hard to implement on the router OS as its not been open for the educational or experimental use and due to restrict access to routers OS by the vendors we can’t show the implementation in real world, that’s why we are lacking the part of implementing the mechanism in this paper, which could only be done by the vendors side who have only the access to reprogram the router’s firmware weather it could be physical or simulated router.

6.2 Altering MTU - By rising MTU size

Since the proposed method seems to violate the main purpose or definition of MTU in the network devices by altering it according to the incoming packet size. Then why shouldn’t the MTU is removed, then to keep on overriding for each larger incoming packet? The proposed method has set a threshold limit of rising the MTU of the forwarding link and this threshold limit helps in keeping the main work of MTU unaffected.

With the increase in link data rate and the new highly equipped modern equipment’s the standard MTU 1500 bytes remain unchanged for decade which is now been in question by prior research carried out in support of using jumbo frames in IPv6 network by [30] shows the increase in network throughput by up to 117% and rising the standard MTU 1500 bytes to 9000 bytes in [31] which found significantly increase in throughput and resistant to the packet drop due to reduce in the number of frame transmitted. In [32] found that that the power consumption is proportional to the link load and with the larger MTUs allows a link to enter a low power mode, thereby improving energy efficiency in wire-line Ethernet.

6.3 Pre-configuration for Implementation

6.4 Defining a Threshold Value

Since the prior work in [32], [31], [30] have advises to increase MTU to 9000 bytes and the current standard MTU is 1500 bytes. The threshold limit can be set between 1500 to 9000 bytes and keeping the standard 1500 MTU undisturbed. The threshold value can be used as the new prescribed standard MTU value for the Ethernet before implementing as standard MTU globally to help the high performance source and intermediate nodes and high link data with higher network performance as the older devices still can’t work better in MTU size greater than 1500 bytes. The proposed method helps in sending high priority mini-jumbo and jumbo packets without disturbing standard MTU 1500 bytes.

6.5 Halting restart - MTU change without restart

In our proposed method the overriding of MTU will lead to connectivity failure as the change in MTU takes place after a automatic or manual restart of the network devices, to overcome this connectivity failure a pre-configuration is needed from vendor side by halting this feature of restarting the routers after a change in MTU and should make the change in MTU takes place without restart. This pre-configuration must be done before the implementation, for the smooth working of the proposed method.

These implementation issues are brought forward only in the view of designing the proposed method much more better by keeping the issues as the main factors in the knowledge of the researches who put their interest on making it more efficient and reliable.
7 Conclusions & Future Work

This paper presents a scheme called DMTU that tries to reduce the packet drops inside the network in order to increase the throughput and decrease the overall packet latency. The algorithm proves to be robust and fast. This Method reduces usage of ICMP PTB messages in parallel Dynamic MTU scheme and nearly has null usage of ICMP PTB messages in Standalone Dynamic MTU. It is completely independent of the Source node and any kind of Error messages until used in parallel with Path MTU Discovery Scheme as done by state-of-art algorithms in the same domain. Unlike PMTUD, the algorithm processes the packet as it arrives and forwards in no time. This paper presents different versions of the DMTU algorithm and shows how the minor optimizations help the algorithm to adapt to different networks. The mathematical and graphical analyses of DMTU algorithm shows its effectiveness as compared to the state-of-the-art PMTUD algorithm.

The DMTU algorithm can be further enhanced in terms of the overhead time it takes to process the packet and overrides the MTU of a node which will have a great effect in the complete implementation and working of the algorithm and hence thereby decreases the time delay further. Our main motive of designing the DMTU algorithm is to get rid of the Packet truncation and congestion in IPv6 protocol networking which effects the Quality of Service of the network.

The further work in finding the time complexity and space complexity of the program that takes to run algorithm program on end-devices and intermediate node.

References

[1] V Fuller and T Li. Classless inter-domain routing (cidr): The internet address assignment and aggregation plan. RFC 4632, IETF, Internet Requests for Comments, 8 2006.

[2] Steve Deering and Robert M. Hinden. Internet protocol, version 6 (ipv6) specification. RFC 8200, IETF, Internet Request for Comments, 7 2017.

[3] Richard Graveman, Pekka Savola, Mohan Parthasarathy, and Hannes Tschofenig. Using ipsec to secure ipv6-in-ipv4 tunnels. RFC 4891, IETF, Internet Request for Comments, 5 2007.

[4] P. Wu, Y. Cui, J. Wu, J. Liu, and C. Metz. Transition from ipv4 to ipv6: A state-of-the-art survey. IEEE Communications Surveys Tutorials, 15(3):1407–1424, 2013.

[5] Robert M. Hinden and Steve Deering. Ip version 6 addressing architecture. RFC 4291, IETF, Internet Requests for Comments, 2 2006.

[6] IEEE. Ieee standards for local area networks: Carrier sense multiple access with collision detection (csma/cd) access method and physical layer specifications. ANSI/IEEE Std 802.3-1985, 1985.

[7] Noor Mast and Thomas J. Owens. A survey of performance enhancement of transmission control protocol (tcp) in wireless ad hoc networks. EURASIP Journal on Wireless Communications and Networking, 2011(1):96, 2011.

[8] R. Khalili and K. Salamatian. A new analytic approach to evaluation of packet error rate in wireless networks. IEEE Proceedings of the 3rd Annual Communication Networks and Services Research Conference (CNSR’05), pages 333–338, 2005.
[9] Buddha Singh and D. K. Lobiyal. A mac-layer retransmission technique for collided packets in wireless sensor network. *Wireless Personal Communications*, 72(4):2499–2518, 2013.

[10] Jack McCann, Stephen Edward Deering, Jeffrey C. Mogul, and Robert M. Hinden. Path mtu discovery for ip version 6. RFC 8201, IETF, Internet Request for Comments, 7 2017.

[11] Christopher A. Kent and Jeffrey C. Mogul. Fragmentation considered harmful. *SIGCOMM Comput. Commun. Rev.*, 25(1):75–87, January 1995.

[12] Maikel De Boer, Jeffrey Bosma, B Overeinder, and Willem Toorop. Discovering path mtu black holes on the internet using ripe atlas. Master’s thesis, University of Amsterdam, The Netherlands, 7 2012.

[13] Jeffrey C. Mogul and Stephen Edward Deering. Path mtu discovery. RFC 1191, IETF, Internet Requests for Comments, 11 1990.

[14] Vijay Kumar Vasantha. Isis: Path mtu calculation in isis. Internet Draft draft-kumar-isis-path-mtu-00.txt, IETF, 8 2008.

[15] Z Hu, Y Zhu, Z Li, and L Dai. Is-is extensions for path mtu draft-hu-lsr-isis-path-mtu-00. Technical report, IETF Internet-Draft, 6 2018.

[16] M. Mathis and J. Heffner. Packetization layer path mtu discovery. RFC 4821, IETF, Internet Requests for Comments, 2007.

[17] Matt Mathis. Raising the internet mtu. *Pittsburgh supercomputing center*, 04 2007.

[18] G. Fairhurst, Tom Jones, M. Tüxen, Irene Rüngeler, and T. Völker. Packetization layer path mtu discovery for datagram transports. RFC 8899, IETF, Internet Requests for Comments, 09 2020.

[19] Tasnuva Mahjabin, Yang Xiao, Guang Sun, and Wangdong Jiang. A survey of distributed denial-of-service attack, prevention, and mitigation techniques. *International Journal of Distributed Sensor Networks*, 13(12):1550147717741463, 2017.

[20] Khaled Elleithy, Drazen Blagovic, Wang Cheng, and Paul Sideleau. Denial of service attack techniques: Analysis, implementation and comparison. *Journal of Systemics, Cybernetics and Informatics*, 3:66–71, 01 2006.

[21] Matthew Luckie, Kenjiro Cho, and Bill Owens. Inferring and debugging path mtu discovery failures. 2005.

[22] S Knowles. Iesg advice from experience with path mtu discovery. RFC 1435, IETF, Internet Requests for Comments, 3 1993.

[23] K Lahey. Tcp problems with path mtu discovery. RFC 2923, IETF, Internet Requests for Comments, 9 2000.

[24] P Savola. Mtu and fragmentation issues with in-the-network tunneling. RFC 4459, IETF, Internet Requests for Comments, 4 2006.

[25] Jon Postel. Internet protocol. RFC 791, IETF, Internet Request for Comments, 9 1981.

[26] P Hoffman, A Sullivan, and K Fujiwara. Dns terminology. RFC 7719, IETF, Internet Request for Comments, 12 2015.
[27] K. Rikitake, H. Nogawa, T. Tanaka, K. Nakao, and S. Shimojo. Dns transport size issues in ipv6 environment. *2004 International Symposium on Applications and the Internet Workshops. 2004 Workshops, IEEE.*, pages 141–145, 2004.

[28] J McCann, S Deering, and J Mogul. Path mtu discovery for ip version 6. RFC 1981, IETF, Internet Requests for Comments, 8 1996.

[29] Ishfaq Hussain and Janibul Bashir. Measuring time delay in path mtu discovery in transmitting a packet in ipv4 and ipv6 network. 2020.

[30] Supriyanto, R. Sofhan, R. Fahrizal, and A. Osman. Performance evaluation of ipv6 jumbogram packets transmission using jumbo frames. *2017 4th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI) [proceedings]*, pages 1–5, 2017.

[31] D. Murray, T. Koziniec, K. Lee, and M. Dixon. Large mtus and internet performance. pages 82–87, 2012.

[32] P. Reviriego, A. Sanchez-Macian, J.A. Maestro, and Chris J. Bleakley. Increasing the mtu size for energy efficiency in ethernet. *IET Irish Signals and Systems Conference (ISSC 2010) [proceedings]*, 06 2010.

**Dr. Janibul Bashir** has received his Ph.D in Computer Science and Engineering from Indian Institute of Technology Delhi in 2019. He is currently working as an Assistant Professor in the Information Technology Department at National Institute of Technology, Srinagar. Before joining NIT Srinagar, he was working as a Software Engineer at Samsung Research India, Bangalore where he was awarded with Spot award at Samsung for developing 10 applications for the Tizen platform (Smartphone platform). He was also awarded with Gold Medal from NIT Srinagar in 2014. He current research is improving the performance of multicore systems, on-chip security, and application of machine learning techniques in the computer architecture domain (emerging technologies, network-on-chip, thermal management). He has also extended research interest in the operating systems and parallel programming (distributed systems). He can be reached at janibbashir@nitsri.ac.in.

**Ishfaq Hussain** was previously a student of National Institute of Technology, Srinagar with field of study concentric to Information Technology. He has a broad area of research interests in computer science and mathematical areas but not concentric to computer network, algorithms, mathematical analysis and computer architectures. He is currently a part of research community by contributing high authentic work to research communities of computer and mathematical society. He can be reached at ishfaqhusain90@gmail.com