

IPv6 Operations (v6ops) Working Group
Internet Draft
Intended status: Informational
Expires: March 2022

E. Vasilenko
X. Xiao
Huawei Technologies
D. Khaustov
Rostelecom
September 17, 2021

IPv6 Oversized Packets Analysis
draft-vasilenko-v6ops-ipv6-oversized-analysis-01

Abstract

The IETF has some initiatives relying on IPv6 Extension Headers added in transit: SRv6, iOAM. Additionally, some recent developments are overlays (SRv6, VxLAN, NV03, L2TPv3, and LISP). It could create oversized packets that need to be dealt with. This document analyzes available standards for the resolution of oversized packet drops.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on February 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

Internet-Draft

Oversized Analysis

August 2021

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the [Trust Legal Provisions](#) and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Terminology and pre-requisite.....	2
2.	Problem statement.....	3
3.	Solutions.....	5
3.1.	Provision links with big enough MTU.....	6
3.2.	Frugal usage of Extension Headers.....	7
3.3.	Fragmentation and reassembly at the tunnel ends.....	9
3.4.	PMTUD by original packet source.....	12
3.5.	Packetization Layer MTU Discovery.....	14
4.	Conclusion.....	15
5.	Security Considerations.....	15
6.	IANA Considerations.....	15
7.	References.....	16
7.1.	Normative References.....	16
7.2.	Informative References.....	18
8.	Acknowledgments.....	19

[1.](#) Terminology and pre-requisite

We do assume good knowledge or frequent references to [\[PMTUD\]](#) and [\[IPv6 Tunneling\]](#). Terminology is inherited from [\[PMTUD\]](#).

Link MTU - the maximum transmission unit, i.e., maximum packet size in octets that can be conveyed over a link.

Path MTU (PMTU) - the minimum link MTU of all links in a path between a source node and a destination node.

Path MTU Discovery (PMTUD) - the process by which a node learns the PMTU of a path.

EMTU_S - Effective MTU for sending; used by upper-layer protocols to limit the size of IP packets they queue for sending.

EMTU_R - Effective MTU for receiving; the largest packet that can be reassembled at the receiver.

Packetization Layer - the layer of the network stack that segments data into packets.

PLPMTUD - Packetization Layer Path MTU Discovery, the method of detecting path MTU at packetization layer, which is an extension of classical PMTU Discovery.

PTB (Packet Too Big) message - an ICMPv6 message reporting that an IPv6 packet is too large to forward through some link.

MSS - the TCP Maximum Segment Size, the maximum payload size available to the TCP layer. This is typically the Path MTU minus the size of the IP and TCP headers.

2. Problem statement

IPv6 is strict regarding fragmentation - it must NOT be done in transit (section 4.5 of [[IPv6](#)]).

IPv6 sees rapid developments in recent years. A lot of additional functionality has been added primarily by adding options to Extension Headers and/or using overlay encapsulation. All of the above expand the packet size. This could lead to oversized packets that would be dropped on some links.

Massive parallelism in traffic delivery is the additional challenge developed in the last 10 years: ECMP on one hop could reach 16 (or even more), which creates the end-to-end possibility for 64k paths on just 5 hops (example from big production network). Different paths could have a different set of Extension Headers and different PMTU as a result. PMTU is effectively becoming dynamic: we could never know how many additional headers would be added at a particular time to the particular packet on the particular path.

The old classical PMTUD problems are still with us: filtered ICMPv6 messages, drops related to Extension Headers before next hop MTU has been evaluated (no Packet Too Big message sent).

Standards have two important numbers that we would need for our discussion:

- o [\[IPv6\]](#) chapter 5 requires that every link should have the MTU of 1280 octets or greater ($2^{10} + 2^8$ - it probably explains the choice of this size)

- o [\[IPv6\]](#) requests minimum EMTU_R (reassembly buffer) in 1500 octets. An upper-layer protocol or application that depends on IPv6 fragmentation to send packets larger than the MTU of a path should not send packets larger than 1500 octets unless it has the assurance that the destination is capable of reassembling packets of that larger size

The reassembly buffer is much above 1500B for the majority of desktop and server OSes. Windows 10 has "Reassemblylimit" almost 64MB (you could look by "netsh interface ipv6 show global"). Different flavors of Linux have "ipfrag_high_thresh" between 256KB and 4MB (you could look by "more /proc/sys/net/ipv4/ipfrag_high_thresh"). iOS has "maxReceiveIPv6BufferSize" 64Kb.

The reassembly is not so good for embedded OSes. From the four primary OSes for IoT (Contiki, FreeRTOS, Mbed OS, MicroPython), only Mbed OS has the capability for 5 fragments by default, and it is possible to activate reassembly on Contiki. In all cases, the buffer is just a few packets of 1280B or 1500B. IoT devices may not be capable to reassemble the packet that the server in the cloud would send to it. Hence, ICMP PTB is still very important for some OSes.

There is only one solution by [\[IPv6\]](#) architecture for the PMTU problem - decrease packet size on the original source. It is workable up to the minimum limit for IPv6 packets (1280B). The typical transit link had MTU not much bigger than 1500B for a long time, only the space for a few additional MPLS labels was reserved. 220B left could be considered as guaranteed for additional functionality in Extension and Encapsulation headers. It could be enough for the next decade if we would make some precautions - see discussion below.

[Huston-2016] and [\[Huston-2021\]](#) did an investigation on a different

topic (fragmentation), but he has good statistics related to MTU drops up to 1500B that did show a 5% drop for MTU as small as 1455B. Additionally, [[Huston-2016](#)] has found the big drop spike (69% from all drops!) at 1480B, 20B less is presumable for IPv6 encapsulation into IPv4. [[Huston-2021](#)] has shown twice bigger fragmentation drop for bigger packets with the peak at 1408 octets, especially for Asia. As you can see - 1500B is not always available now, the reason is not well understood. Hence, we do not have 220B for additional headers in all situations. We could be reasonably optimistic that such a type of tunneling would disappear in the long term. Later, we would stick to an optimistic assumption that 220B is available in most situations. It is still possible to have the more pessimistic estimation (200B? 175B?) looking to Huston's data.

The hungriest protocol known is SRv6 that could add 40B of IPv6 underlay tunnel header (called "outer IP header" in [[SRH](#)]), 16B of SRH header itself, and additionally up to 10 IPv6 addresses in the SID stack (potentially even more). It is already 216B - very close to 220B optimistic limit. It makes the introduction of any additional functionality quite challenging without rigorous expansion of all links to bigger MTU.

Initial SRv6 implementations that trespassed safe limit in 220B are the reason for recent activities in MTU problem research. We see many recent efforts to improve Path MTU Discovery (which would be mentioned in the document) - let us find the rationale behind it.

[3.](#) Solutions

Let's consider first the reassembly buffer problem as the simpler one.

Minimal buffer for packet reassembly (1500B) is potentially possible to increase in new standard updates, but then would be the problem with the transition, because this limitation would be already programmed into billions of IoT hosts - it would need big time to be sure that we do not have old implementation anymore.

There is no good solution for the problem of bloating headers above 220B for hosts. We need to keep headers below the 220B limit for embedded OSes. Fortunately, we are far from this problem yet - very limited additional functionality is implemented directly on the

hosts (like [PMTU by HbH] or APN6). This problem should be looked at again in some number of years, it may be that in the future we would have to increase default EMTU_R on all hosts to give the possibility for new functionality.

Let's now return to our primary problem of not enough PMTU.

There is a low probability that the Internet community would agree to decrease the minimal IPv6 packet size (1280B). Hence, the oversized problem could not be resolved in that direction.

It is possible to partially alleviate the MTU problem in some network zones where all transit nodes have big enough MTU. Transit nodes should delete extension headers before packets would leave "high MTU network zone". The leakage of a big header to a host could overflow EMTU_R buffer. The majority of RFCs recommend carriers delete additional headers before forwarding traffic to the client - this practice should be strictly followed.

The SPRING working group is actively developing a compressed version of SRv6 that should leave space for other functionality, even on current transit routers that sometimes do not support much above 1500B.

All solutions for packet drop avoidance as a result of oversized packets could be classified into 4 classes. They are examined one by one.

[3.1](#). Provision links with big enough MTU

MTU supported by the host's links is typically 1500 Bytes. Backbone link's MTU could be up to 9000 Bytes on modern hardware. PMTUD is not needed in an ideal world.

Reality is not that good:

- o Some old devices still support just a few additional MPLS labels above 1500B on Ethernet. It was historically a problem to cross 1536B because the IEEE specification for 802.3 assumes that a bigger number in the Length field means the Type of the payload.
- o We could have middleboxes that would not support MTU much bigger than 1500B MTU for a long time.

- o Ethernet is very mature now in relation to big MTU support, but that could be a challenge for other link-layer technologies (for example WiFi, satellite links, radio links, etc.).
- o Packet Links could be rented from a third party - no possibility to change the MTU.
- o Big MTU may negatively influence buffer size - see below.
- o The majority of vendors set the default MTU to 1500B (with variations on what is counted inside MTU). It is time-consuming to change the MTU on the production network.
- o Some hosts (especially for storage traffic in Data Centers) could use 2500B or 9000B MTU that challenges the possibility of having always bigger MTU in the backbone.

Cost-optimized equipment architecture (especially used for switches, but applicable for many routers as well) may not split packets in the buffer memory. So small packet would occupy a bigger buffer space reserved for the packet with maximum MTU. This limitation effectively decreases the potential number of packets that could be buffered. Most of the host packets are still limited to 1500B size. MTU 9000B would just lead to wasting buffer memory with an efficiency of 1/6. The average packet size is twice smaller, hence in the worst-case buffer efficiency could be up to 1/12. Buffer memory is about 30% of the router cost. It is not acceptable to

increase buffer memory cost 12 times. Hence, in many cases, it does not make sense to increase MTU to the maximum supported by the switch or router. One should always check with the vendor the impact of using a big MTU on buffering for the particular product. MTU should be increased to the number that is bigger than the maximum MTU expected from hosts + the size of all possible network overhead + underlay IPv6 header (if present). 9000B MTU makes sense in DC, cross-DC environment, or for platforms that fragment packets for smaller sells in the memory.

[MTU issues in Tunneling] [section 3.3](#) discusses the opposite solution: decrease MTU on links to hosts to be sure that a host would always generate small enough MTU for the backbone. This solution was possible for small tunnel overhead, but now we are talking about the situation when 220B margin is not enough.

[L3VPN] and [\[EVPN\]](#) do attach an additional label and could create oversized packets. Still, the MPLS header cannot point to the

original MPLS router that has an attached service label. Additionally, a VPN IP packet could use private address space or no IP address at all (for EVPN). It blocks the possibility to properly organize the PMTUD process. Hence, [[L3VPN](#)] and [[EVPN](#)] have been developed under the assumption that all MTUs on the path would be expanded for at least 8 bytes that are needed for services over the MPLS data plane.

We have recent [Generic Delivery Functions] that may permit fragmentation for MPLS services, but it is a personal draft yet.

[Pseudowire Fragmentation] is the rear case when fragmentation is available over MPLS for one type of service.

[VxLAN] [section 4.3](#) also uses the approach: "it is RECOMMENDED that the MTUs across the physical network infrastructure be set to a value that accommodates the larger frame size due to the encapsulation".

Packet drop statistics and big activity in IETF prove that the PMTUD problem persists.

"Raise MTU on transit" is the best solution, if it is available.

[3.2](#). Frugal usage of Extension Headers

Some new functionality (especially source routing with a big SID stack) could decrease headers size without a big loss of functionality (for example, use loose node appointment in SID

stack). Some functionality (like iFIT or iOAM) could be completely omitted in the situation that would lead to packet drop. It is effectively "the tradeoff of functionality to PMTU control".

The important point here is that the transit node attaching an additional header should be aware of all MTUs along the assumed packet path to predict how big MTU is still acceptable.

[PMTUD] is readily available for tunneling interfaces - tunnel source should be aware of PMTU of the tunnel (by PTB feedback messages). But we have cases when it is not enough:

- o SDN controller (or management system in general) could assist in

provisioning of extension headers (including SFC, iOAM, BIER) and encapsulation headers (SRv6, VxLAN) - should be the way to report MTUs to Controller.

- o ICMPv6 PTB would be directed to the transit control plane only in the case of problems inside the tunnel. PTB messages from outside of the tunnel would be directed to the source node. It is difficult to snoop PTB on transit nodes.

Hence, we see many initiatives to collect and manage MTU by many popular protocols for routing and traffic engineering: [PMTU by ISIS], [PMTU by BGP-LS], [PMTU by PCEP], [PMTU by SR-Policy].

Moreover, these protocol extensions would become even more useful in the future when it would not be possible to squeeze all extension headers into 220B anymore. Frugal attachment of new headers on transit nodes would increase the need for awareness of PMTU - it should stimulate MTU collection by all other popular protocols (OSPF, normal BGP on peering borders).

This approach has a fundamental problem: full knowledge about all MTUs in the domain could not help to estimate the real path for a packet, because of massive ECMP used by many networks (at least by all Carriers). Non-routing protocols do not have a proper engine to estimate traffic paths and predict PMTU as well. And even more, if L2 ECMP is used or some links are rented from another carrier it will again be impossible to predict the exact path and the PMTU.

The second problem of this approach could be classified as "chicken and egg". We already have a much better solution for MTU drop - increase MTU (see the previous section). We are looking for other solutions only because upgrading equipment (to better MTU) is not possible for some reason. But new protocols introduction would also demand equipment upgrade and thus making frugal headers less

valuable. However, an upgrade for the control plane should be cheaper than an upgrade for the data plane, if the vendor would support such an approach.

Hence, the solution discussed in this section has only limited applicability.

[3.3](#). Fragmentation and reassembly at the tunnel ends

The tunnel source behaves like a host with respect to the tunnel header. It is possible to properly adjust PMTU for the tunnel by [\[PMTUD\]](#), so it is potentially possible to fragment all packets bigger than PMTU.

[IP Encapsulation] is the earliest standard for IP-in-IP encapsulation. [Section 5.1](#) discusses that it is possible to fragment IP packets before tunnel encapsulation, so there is no need to reassemble packets on the other tunnel end - reassembly could happen on the destination host. It does not have additional cost implications on tunnel ends. This approach did work for IPv4 in the case of the "don't fragment" bit cleared. It fully contradicts IPv6 architecture that does not permit to fragment packets on transit - no standard has risked proposing such a solution for IPv6.

Some standards do propose IPv6 fragmentation (primarily for packets 1280B and below), but fragmentation is recommended after encapsulation. It would lead to packet reassembly on the other tunnel end to hide (from destination host) the fact of transit fragmentation. It does minimize IPv6 architecture disruption.

Many standards discussed below ([MPLS Encapsulation], [\[L2TPv3\]](#), [\[VxLAN\]](#), [\[NV03\]](#)) forgot to mention that packets 1280 and below should be fragmented. This inaccuracy did not create any problem in production networks because we typically have 220B for all headers - it is big enough for many tunnels nested into each other. The situation could change in the next years because of Extension Headers expansion by different functions. It could create pressure to return to many mature standards and clarify the situation: what to do when the 1280B packet could not go through the tunnel.

The Fragmentation has a few issues that make it not popular:

- o Fragmentation could double buffer requirements (we assume split only in 2 fragments). We could ignore small additional buffer requirements for packets that may be lost and need to wait some time before reassembly, the Internet is not productive anyway

after a few percentages of packet drops. The buffer memory is about 30% of the router cost. A 30% cost increase would not be accepted by the majority of owners. Albeit, some middleboxes

already have enough buffer memory that could be reused for packet reassembly.

- o In general, IPv6 architecture does not approve fragmentation in transit in all standards (except the recent draft [IP Tunnels] - see below). [PMTUD] [section 5.1](#): "packetization layers are encouraged to avoid sending messages that will require fragmentation".
We would discuss in this section some situations when tunnel fragmentation is inevitable.
- o [Fragile Fragmentation] has a good collection of all problems related to fragmentation (additionally to the above: breaks ECMP, stateful processing, policy routing, and has many security attack vectors). [Fragile Fragmentation] strongly recommends avoiding fragmentation, but not deprecate yet.

The primary RFC for tunneling is [IPv6 Tunneling] - it is the oldest standard that was later reused by many other standards (including the latest SRH). It permits fragmentation only for the case when the original packet is already minimal (1280B or less) - see [section 7.1](#). It mandates dropping the packet and signaling ICMPv6 PTB to the source (request to decrease the PMTU size at the source) for all other cases.

[MPLS Encapsulation] [Section 5.1](#) has the name: "Preventing Fragmentation and Reassembly". It does stress again: "IPv6 intermediate nodes do not perform fragmentation in any event".

[L2TPv3] [section 4.1.4](#) has a similar comment: "Note that IPv6 does not support "in-flight" fragmentation of data packets".

[VxLAN] [section 4.3](#) is strict: "VTEPs MUST NOT fragment VXLAN packets."

[NV03] [section 4.4.4](#) is strict too: "It is strongly RECOMMENDED that Path MTU Discovery ([PMTUD]) be used to prevent or minimize fragmentation."

[IPv6 GRE] [section 3.3](#) does recommend fragmentation only for packets that are less than 1280B.

The most recent draft for all types of tunnels is [IP Tunnels]. It is already referenced by many IETF documents. It is complicated to cover all use cases (any IP over any IP in any situation), but the

net result is: much bigger part of the traffic proposed to be fragmented into the tunnel. [Section 3.3](#): "The path between ingress and egress interfaces has a path MTU, but the endpoints can exchange messages as large as can be reassembled at the destination (egress interface), i.e., the EMTU_R of the egress interface".

A short explanation of proposed functionality: original host would try to transmit biggest flows (by volume) on maximum PMTU, that tunnel source would not try to correct by PTB messages up to 1500B. Hence, the tunnel source would not have any option except to fragment. The principal problem here is the absence of PTB messages for the packet size between PMTU and statically appointed EMTU_R. Let's see how it has been formulated in more detail.

[IP Tunnels] introduces a new variable "Tunnel MTU" that should not change as a result of PMTUD. The procedure to change "Tunnel MTU" is out of the draft discussion – it is pushed to specifications of particular tunnels in the last paragraph of [section 4.2.2](#). Moreover, it is even assumed that PLPMTUD could be used on the router for "Tunnel MTU" discovery because this parameter is considered as an above network layer (like transport layer on the host). Separate [section 4.2.3](#) is dedicated to the explanation that the newly introduced "Tunnel MTU" cannot be adjusted dynamically. There is a recommendation for the default "Tunnel MTU": typical host EMTU_R (1500B) minus tunnel outer headers overhead. The good question could be: if it is so difficult to manage "Tunnel MTU" dynamically, then why this variable was introduced?

The MTU of the tunnel is renamed into MAP (maximum atomic packet), MAP should be corrected by PMTUD feedback from inside the tunnel. [Section 4.2.2](#) states that everything up to "Tunnel MTU" should be accepted to the tunnel, one long packet (with inner and outer headers) should be created. Then it should be split into fragments below MAP size.

Initially, "tunnel MTU" and MAP could be manually synchronized by the administrator (with the difference in tunnel overhead). But any additional overhead on the tunnel path (nested tunnel, smallest Extension Header) would result in PMTUD that decreases MAP, but would not change "Tunnel MTU". It would turn on fragmentation for all bulk traffic. This situation is quite probable now (see [Huston-2020] on MTUs available on the Internet) and it would be even more probable in the future when many additional extension headers would be used. Hence, the requirement in [section 5.3.1](#) "do NOT try to deprecate fragmentation" is indeed important.

[Section 3.6](#) has the same approach as all other standards to the question of when fragmentation should happen: "this document assumes that only outer fragmentation is viable because it is the only approach that works for both IPv4 datagrams with DF=1 and IPv6". a considerable increase in fragmentation is proposed for the reasons

Internet-Draft

Oversized Analysis

August 2021

of academic purity: the router part of the router should behave as a router, the host part of the router should behave as a host without any deviations.

Additional fragmentations would create all of the problems discussed in [Fragile Fragmentation] and substantially increase the cost of tunnel endpoints. There is a high probability that draft [IP Tunnels] would be rejected by the market for cost reasons.

Additionally, we should point that statistics for fragmented packet drop on the Internet is still high enough (7%) - see [[Huston-2021](#)].

Fragmentation is the least probable solution for oversized packet drops.

[3.4.](#) PMTUD by original packet source

[PMTUD] is mandatory in IPv6 architecture, because IPv6 does not have fragmentation in transit. We could see recommendations in many RFCs not to block ICMPv6 PTB completely (it could be rate-limited - see [[ICMPv6](#)] [section 2.4](#)). [[DPLPMTUD](#)] [section 1.1](#) has a very good collection of reasons why PTB message may not be delivered to the source - it is used as justification to augment PMTUD by [[DPLPMTUD](#)].

We should not see this problem for all non-tunneling protocols in the majority of environments. ICMPv6 PTB should be delivered to packet source, packet source would dynamically decrease PMTU to adapt to new realities. PMTU could change dynamically because some transit nodes could introduce additional extension header ad-hoc or ECMP could switch flow to a different path.

[IPv6 Tunneling] mandates to relay ICMPv6 PTB by tunnel ends for ICMPv6 messages received from the inside tunnel. [IPv6 Tunneling] does not use "relay" terminology, but [section 8](#) explains in detail how to reconstruct and retransmit ICMP messages to the original packet source (delete all tunnel-related information).

[MTU issues in Tunneling] [section 3.2](#) discusses the same approach. [[L2TPv3](#)] [section 4.1.4](#) refers to the [IPv6 Tunneling]. We could assume it as the request for PTB messages relay too.

[[SRH](#)] [section 5.4](#) confirms full adherence to ICMPv6 PTB relay approach: "For IP packets encapsulated in an outer IPv6 header, ICMP error handling is as defined in [IPv6 Tunneling]".

[VxLAN] [section 4.3](#) proposes to use PMTUD: "Path MTU discovery MAY

be used to address this requirement as well".

[NV03] [section 4.4.4](#) assumes PMTUD too: "It is strongly RECOMMENDED that Path MTU Discovery ([\[PMTUD\]](#)) be used to prevent or minimize

fragmentation".

[IPSec] [section 8.2.1](#) requests that PMTU should be maintained for the tunnel and signaled to the original packet source as soon as any new packet would arrive.

[IPv6 GRE] [section 3.3](#) clearly instructs developers to drop the oversized packets and send PTB for packets bigger than tunnel MTU. The method of PMTU detection is fully IPv6 compliant: "the GMTU is equal to the PMTU associated with the path between the GRE ingress and the GRE egress, minus the GRE overhead".

[MPLS Encapsulation] [section 5.1](#) specifies the same approach: tunnel head-end should use [\[PMTUD\]](#) to understand tunnel MTU, then "the packet will have to be discarded, but the tunnel head should send the IP source of the discarded packet the proper ICMP error message".

[VxLAN], [NV03], [IPSec], [IPv6 GRE], and [MPLS Encapsulation] do not request for tunnel endpoint to relay PTB messages. PMTUD should be used to set proper MTU for the tunnel, then subsequent packets could trigger PTB messages to the packet source. It would create an additional round trip delay compared to the original [IPv6 Tunneling] relay approach for the first PTB message. This small deficiency could be partially explained by the desire of many standards to be universal for IPv6 as well as IPv4. As a reminder, IPv4 may not have enough information in the ICMP message to properly reconstruct a relay message (64bits of source packet by [RFC 792](#)).

[IP Tunnels] is the only draft that contradicts [IPv6 Tunneling] (and every other protocol based on top) - it does clearly prohibit relay PTB messages. It states in [section 3.3](#): "When such messages (PTB) arrive at the ingress interface ("ingress interface" is the tunnel interface in this draft), they may affect the properties of that interface (e.g., its MTU), but they should never directly cause new ICMPs in the outer network". This idea is generalized in [section 5.1](#) as "ICMP messages MUST NOT be generated by the tunnel (as a link)". The motivation assumed in the draft is to fully mimic host behavior on the router virtual (tunnel) interface, because the host would not retranslate PTB messages.

We see that "Flow Label" is gaining popularity. [IPv6 Tunneling] and [ICMPv6] do not have strong recommendations for "Flow Label" - it was not an important topic at that time. The only small improvement that makes sense to do for [IPv6 Tunneling] is to recommend coping "Flow Label" from source packet to tunnel packet and from source packet to ICMPv6 PTB message. It would permit to properly load balance PTB messages to the same path as original traffic - see the problem [ICMPv6 PTB in ECMP] about hash-based load balancing between

many hosts. Copy "Flow Label" to PTB message would not contradict neither IPv6 architecture nor any RFC - it is not mandatory to develop a special standard update for it.

[MTU issues in Tunneling] [section 3.2](#) has a concern that in the case of Lawful Intercept additional encapsulation could produce PTB messages that would show the fact to the monitored host. It is not a very realistic concern, because PMTU could change for many other reasons (especially with the proliferation of new protocols). If it is still a concern, then it makes sense to use another solution for this case: bigger MTU (better) or even fragmentation.

[MTU issues in Tunneling] [section 3.2](#) raises the question about the applicability of "MSS Clamping". The transit node could snoop the transport layer and change MSS exchanged between nodes. This "hack" is not recommended because it breaks the layered model of IETF or OSI.

[PMTUD] is the only mechanism that is universal for all cases and fully compliant with IPv6 architecture. Vendors just need to use it, despite some challenges to relay PTB messages on tunnel ends. Moreover, it makes sense to standardize the relay of PTB messages on tunnel ends - it would improve PMTUD time on original traffic sources for round trip time.

[IPv6] RFC: "It is strongly recommended that IPv6 nodes implement Path MTU Discovery [[PMTUD](#)]".

[3.5](#). Packetization Layer MTU Discovery

[PLPMTUD] and [[DPLPMTUD](#)] have been greatly developed in recent years. Packetization Layer (UDP/TCP) (1) has much more visibility (could see the size of transport layer buffers); (2) could operate under the absence of ICMPv6 PTB (too much filtering); (3) could be

very granular (per-flow). It does have its use cases.

Albeit, PLPMTUD/DPLPMTUD have their restrictions as they: (1) are not universal for all transport protocols; (2) need more resources from the host; (3) are challenging to share PMTU information between applications; (4) need much more round trip times to find suitable PMTU; (5) do not work well on congested paths (difficult to understand the reason for packet loss).

Hence, PLPMTUD is not a replacement for PMTUD - both are needed. As a reminder from [\[PLPMTUD\]](#): "Packetization Layer Path MTU Discovery (PLPMTUD) is most efficient when used in conjunction with the ICMP-based Path MTU Discovery".

PLPMTUD could play as a replacement for PMTUD in the worst-case scenario (ICMP is filtered). It would lead to the original host PMTU decrease too. PLPMTUD could be considered as a redundancy mechanism for PMTUD.

Strictly speaking, [PMTU by HbH] is a network layer mechanism, not a packetization layer. It is mentioned in this section because its usage is very similar to PLPMTUD, [PMTU by HbH] could be considered to some degree as the extension to PLPMTUD. It is not expected to principally change the conclusions of this document.

[4.](#) Conclusion

It is better not to have a problem with oversized packets in the first place. One should upgrade all links to a bigger MTU, if possible.

The host could have MTU as big as transit node. It would be never possible to deprecate PMTUD. It is important to follow the recommendations of [\[PMTUD\]](#) and [IPv6 Tunneling] for ICMPv6 PTB message delivery to the original traffic source. Tunnel sources should perform the relay function to make sure that the original traffic source would get the PTB message faster.

The temporary 220B limit for all headers pushes us to the frugal implementation of new extension headers. This limit would be alleviated after all backbone links would be upgraded to a much bigger MTU than 1500B. Additional protocols to collect MTU

information could help in the transition period to attach additional headers frugally. It is true for all new protocols: SRv6, SFC, BIER, iOAM, APN6.

[PLPMTUD] and [DPLPMTUD] are not the replacement for [PMTUD], but could help in some scenarios.

Fragmentation is not at all a solution for oversized packet drops.

[5. Security Considerations](#)

[PMTUD], [PLPMTUD], [DPLPMTUD], and [Fragile Fragmentation] have some attack vectors discussed. This document does not introduce additional security vulnerabilities.

[6. IANA Considerations](#)

This document has no request to IANA.

Vasilenko

Expires February 25, 2022

[Page 15]

Internet-Draft

Oversized Analysis

August 2021

[7. References](#)

[7.1. Normative References](#)

[IPv6] S. Deering, R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", [RFC 8200](#), DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.

[ICMPv6] A. Conta, S. Deering, M. Gupta, "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", [RFC 4443](#), DOI 10.17487/RFC4443, March 2006, <<https://www.rfc-editor.org/info/rfc4443>>.

[PMTUD] J. McCann, S. Deering, J. Mogul, R. Hinden, "Path MTU Discovery for IP version 6", [RFC 8201](#), DOI 10.17487/RFC8201, July 2017, <<https://www.rfc-editor.org/info/rfc8201>>.

[IPv6 Tunneling] A. Conta, S. Deering, "Generic Packet Tunneling in IPv6 Specification", [RFC 2473](#), DOI 10.17487/RFC2473, December 1998, <<https://www.rfc-editor.org/info/rfc2473>>.

[ICMPv6 PTB in ECMP] M. Byerly, M. Hite, J. Jaeggli, "Close

Encounters of the ICMP Type 2 Kind", [RFC 7690](#), DOI 10.17487/RFC7690, January 2016, <<https://www.rfc-editor.org/info/rfc7690>>.

[MTU issues in Tunneling] P. Savola, "MTU and Fragmentation Issues with In-the-Network Tunneling", [RFC 4459](#), DOI 10.17487/RFC4459, April 2006, <<https://www.rfc-editor.org/info/rfc4459>>.

[IP Tunnels] J. Touch, M. Townsley, "IP Tunnels in the Internet Architecture", [draft-ietf-intarea-tunnels-10](#) (work in progress), September 2019.

[IP Encapsulation] C. Perkins, "IP Encapsulation within IP", [RFC 2003](#), DOI 10.17487/RFC2003, October 1996, <<https://www.rfc-editor.org/info/rfc2003>>.

[IPSec] S. Kent, K. Seo, "Security Architecture for the Internet Protocol", [RFC 4301](#), DOI 10.17487/RFC4301, December 2005, <<https://www.rfc-editor.org/info/rfc4301>>.

Vasilenko

Expires February 25, 2022

[Page 16]

Internet-Draft

Oversized Analysis

August 2021

[IPv6 GRE] C. Pignataro, R. Bonica, S. Krishnan, "IPv6 Support for Generic Routing Encapsulation (GRE)", [RFC 7676](#), DOI 10.17487/RFC7676, October 2015, <<https://www.rfc-editor.org/info/rfc7676>>.

[MPLS Encapsulation] T. Worster, Y. Rekhter, E. Rosen, "Encapsulating MPLS in IP or Generic Routing Encapsulation (GRE)", [RFC 4023](#), DOI 10.17487/RFC4023, March 2005, <<https://www.rfc-editor.org/info/rfc4023>>.

[L2TPv3] J. Lau, M. Townsley, I. Goyret, "Layer Two Tunneling Protocol – Version 3 (L2TPv3)", [RFC 3931](#), DOI 10.17487/RFC3931, March 2005, <<https://www.rfc-editor.org/info/rfc3931>>.

[VxLAN] M. Mahalingam, D. Dutt, K. Duda, P. Agarwal, L. Kreeger, T. Sridhar, M. Bursell, C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying

Virtualized Layer 2 Networks over Layer 3 Networks", [RFC 7348](#), DOI 10.17487/RFC7348, August 2014, <<https://www.rfc-editor.org/info/rfc7348>>.

[NV03] J. Gross, I. Ganga, T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation", [RFC 8926](#), DOI 10.17487/RFC8926, November 2020, <<https://www.rfc-editor.org/info/rfc8926>>.

[L3VPN] E. Rosen, Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", [RFC 4364](#), DOI 10.17487/RFC4364, February 2006, <<https://www.rfc-editor.org/info/rfc4364>>.

[EVPN] A. Sajassi, R. Aggarwal, N. Bitar, A. Isaac, J. Uttaro, J. Drake, W. Henderickx, "BGP MPLS-Based Ethernet VPN", [RFC 7432](#), DOI 10.17487/RFC7432, February 2015, <<https://www.rfc-editor.org/info/rfc7432>>.

[Huston-2016] Huston, G., "Fragmenting IPv6", Blog Post, 2016, <<https://blog.apnic.net/2016/05/19/fragmenting-ipv6/>>.

[Huston-2021] Huston, G., "IPv6 Fragmentation Loss", Article, 2021, <<https://www.potaroo.net/ispcol/2021-04/v6frag.html>>.

[Fragile Fragmentation] R. Bonica, F. Baker, G. Huston, R. Hinden, O. Troan, F. Gont, "IP Fragmentation Considered Fragile", [RFC 8900](#), DOI 10.17487/RFC8900, September 2020, <<https://www.rfc-editor.org/info/rfc8900>>.

Vasilenko

Expires February 25, 2022

[Page 17]

Internet-Draft

Oversized Analysis

August 2021

[7.2](#). Informative References

[PLPMTUD] M. Mathis, J. Heffner, "Packetization Layer Path MTU Discovery", [RFC 4821](#), DOI 10.17487/RFC4821, March 2007, <<https://www.rfc-editor.org/info/rfc4821>>.

[DPLPMTUD] G. Fairhurst, T. Jones, M. Tuexen, I. Ruengeler, T. Voelker, "Packetization Layer Path MTU Discovery for Datagram Transports", [RFC 8899](#), DOI 10.17487/RFC8899, March 2020, <<https://www.rfc-editor.org/info/rfc8899>>.

[SRH] C. Filsfils, D. Dukes, S. Previdi, J. Leddy, S. Matsushima, D. Voyer, "IPv6 Segment Routing Header (SRH)", [RFC 8754](#), DOI

10.17487/RFC8754, March 2020, <<https://www.rfc-editor.org/info/rfc8754>>.

[PMTU by HbH] R. Hinden, G. Fairhurst, "IPv6 Minimum Path MTU Hop-by-Hop Option", [draft-hinden-6man-mtu-option-02](#) (work in progress), July 2019.

[PMTU by ISIS] Z. Hu, Y. Zhu, Z. Li, L. Dai, "IS-IS Extensions for Path MTU", [draft-hu-lsr-isis-path-mtu-00](#) (work in progress), June 2018.

[PMTU by PCEP] S. Peng, C. Li, L. Han, "Support for Path MTU (PMTU) in the Path Computation Element (PCE)communication Protocol (PCEP)", [draft-li-pce-pcep-pmtu-03](#) (work in progress), October 2020.

[PMTU by BGP-LS] Y. Zhu, Z. Hu, G. Yan, J. Yao, "BGP-LS Extensions for Advertising Path MTU", [draft-zhu-idr-bgp-ls-path-mtu-05](#) (work in progress), November 2020.

[PMTU by SR-Policy] C. Li, Y. Zhu, A. Sawaf, Z. Li, "Segment Routing Path MTU in BGP", [draft-li-idr-sr-policy-path-mtu-03](#) (work in progress), November 2019.

[Generic Delivery Functions] Z. Zhang, R. Bonica, K. Kompella, "Generic Delivery Functions", [draft-zzhang-intarea-generic-delivery-functions-01](#) (work in progress), April 2021.

[Pseudowire Fragmentation] A. Malis, M. Townsley, "Pseudowire Emulation Edge-to-Edge (PWE3) Fragmentation and Reassembly", [RFC 4623](#), DOI 10.17487/RFC4623, August 2006, <<https://www.rfc-editor.org/info/rfc4623>>.

[8.](#) Acknowledgments

Thanks to v6ops working group for problem discussion

Authors' Addresses

Eduard Vasilenko
Huawei Technologies

17/4 Krylatskaya st, Moscow, Russia 121614

Email: Vasilenko.Eduard@huawei.com

Xiao Xipeng

Huawei Technologies

205 Hansaallee, 40549 Dusseldorf, Germany

Email: Xipengxiao@huawei.com

Dmitriy Khaustov

Rostelecom

13/2 Nikoloyamskaya st, Moscow, Russia 109240

Email: Dmitriy.Khaustov@rt.ru