Retrospective on “Fragmentation Considered Harmful”

Jeffrey C. Mogul
jeffmogul@acm.org

Christopher A Kantarjiev
cak@acm.org

This article is an editorial note submitted to CCR. It has NOT been peer reviewed.
The authors take full responsibility for this article’s technical content. Comments can be posted through CCR Online.

ABSTRACT

We look back at our 1987 paper, “Fragmentation Considered Harmful,” to explain why we wrote it, how the prevalence of fragmentation was reduced by approaches such as Path MTU Discovery, and how fragmentation-related issues still lurk in today’s Internet. Our paper listed several reasons why we thought fragmentation was harmful; some were more true in 1987 than they are today, and after our paper was published, the community realized that fragmentation (and the mechanisms used to mitigate it) exposed harms we did not anticipate in our paper.

1 FRAGMENTATION IN IPV4

By 1987, the introduction of Internet protocols (IP and TCP or UDP) had given us the ability to send packets over an end-to-end path that traversed multiple individual networks, often based on different WAN or LAN technologies. Because each of these technologies had been designed for somewhat different environments, they tended to support different *Maximum Transmission Units* (MTUs), defined as the maximum number of bytes carried by a layer-2 (L2) packet. For example, Ethernet originally supported an MTU of 1500 bytes; the ARPANet’s MTU was 1006 bytes. (RFC1191 [2] includes a table of about 15 different MTUs in use in 1990.)

Generally, a host sending a packet does not know the precise end-to-end path that the packet will take over an IP network, and in particular does not know what L2 technology is in use except for the first (directly-connected) link. Therefore, it could send an IP packet that is short enough for the first hop, but too long for one or more of the subsequent hops. This created a challenge that the designers of the original Internet Protocol (IPv4) solved by allowing intermediate routers (often, at that time, called “gateways”) to *fragment* an IP datagram – to break it into several smaller packets, each with its own IP headers. The ultimate receiver can then reassemble the fragments to recover the original datagram – assuming that the sender assigned a unique 16-bit “IP ID” value in the original packet headers, and assuming that no fragments are lost along the way.

Note that TCP inherently tries to avoid fragmentation, by negotiating a Maximum Segment Size (MSS) that typically is the min of the MTUs at the sender and receiver (minus IP/TCP headers). But if an intermediate link has a smaller MTU, that MSS would be too large to traverse the network without fragmentation.

When we wrote “Fragmentation Considered Harmful” [5] in 1987, fragmentation worked well – much of the time. But sometimes it failed, and after debugging some of these failures, we realized that IP’s end-to-end fragmentation might not have been the best solution to the challenge of mismatched MTUs. Instead, we argued that senders should attempt to send their original packets at the largest size that could reliably be carried end-to-end without fragmentation – later work [2] called this the “path MTU,” although our 1987 paper did not use that term.

Jeff Mogul is currently affiliated with Google LLC. This article represents his own opinion and does not represent the views of Google.

2 CHALLENGES IN 1987

In that paper, we identified several specific problems (and a few others not worth mentioning):

- **CPU and memory costs at receivers for reassembly**: While creating fragments at an intermediate router is typically deterministic, a receiver cannot rely on all fragments arriving in order, and so reassembly adds CPU overhead to inspect the fragments and arrange them in order, including a data structure that supports lookups based on the IP IDs of incoming fragments. Also, if some fragments are lost *en route*, buffers could be tied up for many seconds, until the receiver times out and discards the partially-reassembled packet. (Remember that Van Jacobson’s classic paper “Congestion avoidance and control” [4] would not be published until the next year, and so congestive packet loss was a frequent problem.) This added memory pressure; typical systems only had a few MB of DRAM, and not much could be spared for network buffers.
- **IP ID wrap-around**: Successful reassembly depends on matching the 16-bit IP ID field for all of the fragments in an IP packet. The ID must be unique for all packets “in flight” between two hosts, or else fragments from two different packets could be accidentally spliced. We assumed a maximum Time to Live (TTL) of 32 seconds, so a path with fragmentation could support at most $2^{16}/32 = 2048$ packets/sec. We predicted that this rate would become feasible within 5 years, at which point fragmentation would create an artificial rate limit; we are not sure that prediction came true on schedule.
- **Deterministic fragment loss**: The biggest problem, and the one that prompted us to look into fragmentation in the first place, was that various circumstances could cause deterministic loss of the N^{th} fragment of a packet, so even if the sender timed out and retransmitted, the same fragment would get lost the next time. Remember that if any fragments are lost, the receiver cannot reassemble the original packet – the packet is lost, and the receiver’s IP layer has no way to report that loss back to the sender. Even if the deterministic loss is probabilistic rather than certain (the N^{th} fragment is lost with probability < 1 but larger than the average loss rate), we showed that goodput could be painfully low.

What circumstances caused deterministic packet loss? First, many NICs of the day could not buffer more than a few packets, and if fragments arrived faster than the host could service them, the NIC would have to drop packets. (Operating systems of that era almost always took one interrupt per packet.) Second, given the small amount of host buffering available, the fragments of a large (many-fragment) packet could consistently arrive in a way that overran the available space, requiring the IP layer to drop at least some fragments before the packet could ever be re-assembled – and the remaining fragments took a while to time out, tying up space for no good purpose and exacerbating the deterministic loss.

We argued that sending packets no larger than the path MTU, and relying on TCP's loss-recovery mechanisms, would provide much better results. However, while we suggested several methods to learn the path MTU, in the paper we reported no experimental results for any of those methods, although clearly we had had some operational success by artificially limiting our TCP senders to a 576-byte MTU. Subsequently, the IETF MTU Discovery Working Group standardized (RFC1191 [2]) an MTU-learning method suggested by Geof Cooper, which we mentioned in our paper while advocating other, more complex schemes.

3 HOW THINGS HAVE CHANGED

Today, the specific challenges that we worried most about in 1987 (especially deterministic fragment loss and lack of reassembly buffer space) are mostly non-issues, due to much larger memories, to better-engineered network equipment that can usually handle minimal-size packets at line rate, and probably most of all, to the use of Path MTU Discovery (PMTUD) to avoid most IPv4 fragmentation – and IPv6's non-support of end-to-end fragmentation.

But we still have some challenges related to MTUs, including:

- **PMTUD failure**: Senders using PMTUD transmit all of their packets with the "Don't Fragment" (DF) bit set in the IPv4 header. When such a packet reaches a router that would have to fragment it, instead we expect the router to reply with an ICMP message of type "Destination unreachable" with a code of "fragmentation needed and DF set" – aka the "Datagram too big" message – allowing the sender to detect the failure and, via a heuristic, reduce the packet size and try again. Unfortunately, some router-like systems (including one sold by our own employer at the time) would drop the packet without sending the datagram-too-big message. In other cases, a firewall near the sender would drop the datagram-too-big message, since ICMP messages were often DoS-attack vectors.

  By 2010, Luckie and Stasiewicz [6] reported measurements showing that the PMTUD failure rate was "between 5% and 18%, depending on the MTU of the constraining link," and that many of the remaining failures could be avoided by fixing a few implementation bugs.
- **DoS attacks**: While many senders use PMTUD to ensure successful communication, the Internet does not *require* the DF header bit to be set for IPv4 packets. Therefore, a malicious sender can send a series of fragmented packets which omit one of the fragments from each packet. The receiving victim cannot reassemble any of these incomplete packets, so its reassembly buffers are tied up with bogus packets until these time out. Also, some systems pre-allocate memory based on the total datagram size indicated in the first fragment, so an attacker can send a stream of "first fragments" with unique IP IDs and maximal total-length fields, tying up lots of memory. Of course, a receiver can always limit its own reassembly buffer space and its reassembly timeout, but this can reduce its ability to handle legitimately fragmented packets.
- **Other attacks**: Several other attacks exploit IP fragmentation. For example, an attacker can send several fragments with the same IP-ID that overlap in the reassembly buffer; some older versions of Linux and Windows handle this case incorrectly, and could crash (the "teardrop attack"). Overlapping fragments can also sometimes be used to sneak malicious payloads past intrusion-detection systems. Another attack delivered a longer packet than indicated in the original total-length field, which could also cause kernel crashes.
- **Router overheads**: Typically routers cannot perform fragmentation in their efficient hardware fast paths, so packets needing to be fragmented must invoke a slow path, often in software.

4 SUBSEQUENT RESEARCH

A variety of papers started from where we left off, and proposed other approaches related to fragmentation.

Chandranmenon and Varghese [1] (1998) suggested that avoiding fragmentation by sending packets that fit the Path MTU was undesirable, because larger packets are more efficient, and because a sender using IP multicast would have to use the lowest PMTU in the entire multicast tree. They proposed a variant of hop-by-hop reassembly, called "Graceful Intermediate Reassembly" (GIR), to optimize the use of each link; the use of "Dynamic Segment Sizing" that reduces the TCP segment size only when TCP detects losses; and a more efficient reassembly algorithm. They found substantial performance improvements in their experiments, but to our knowledge, GIR was never deployed widely.

Gilad and Herzberg [3] (2011) reported several novel security attacks exploiting various aspects of the IPv4 and IPv6 fragmentation mechanisms, including "DoS, interception and modification attacks by a blind (spoofing-only) attacker" as well as several other attacks that require some third-party assistance. They wrote "Like many or most previous attacks on IP, our attacks exploit weaknesses in IP's fragmentation mechanisms," which we had not anticipated in 1987.

5 SUMMARY

When we wrote our 1987 paper, we were not sure whether the problem of IP fragmentation was significant enough to be accepted by SIGCOMM, and in hindsight, the lack of any validation of our proposed solutions would probably cause the paper to be rejected by the 2019 SIGCOMM TPC. However, we believe that our work – and a lot of subsequent implementation, validation, and measurement work by many others – did end up making the Internet more successful. Sometimes one need not write a lot of code or a complex algorithm to have an impact.

[1] Girish P. Chandranmenon and George Varghese. 1998. Reconsidering Fragmentation and Reassembly. In *Proceedings of the Seventeenth Annual ACM Symposium on Principles of Distributed Computing (PODC '98)*. 21–29.

[2] Dr. Steve E. Deering and Jeffrey Mogul. 1990. Path MTU discovery. RFC 1191. (Nov. 1990).

[3] Yossi Gilad and Amir Herzberg. 2011. Fragmentation Considered Vulnerable: Blindly Intercepting and Discarding Fragments. In *Proceedings of the 5th USENIX Conference on Offensive Technologies (WOOT'11)*. USENIX Association, 2–2.

[4] V. Jacobson. 1988. Congestion Avoidance and Control. In *Symposium Proceedings on Communications Architectures and Protocols (SIGCOMM '88)*. 314–329.

[5] C. A. Kent and J. C. Mogul. 1988. Fragmentation Considered Harmful. In *Proceedings of the ACM Workshop on Frontiers in Computer Communications Technology (SIGCOMM '87)*. 390–401.

[6] Matthew Luckie and Ben Stasiewicz. 2010. Measuring Path MTU Discovery Behaviour. In *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement (IMC '10)*. 102–108.