# MTU, fragmentation and your local IXP

Apollon Oikonomopoulos
apollon@skroutz.gr — @apoikos

GRNOG 16 — 2024-04-19

skroutz

# About me

- Mechanical engineer by training, sysadmin by trade
- CIO @ skroutz.gr
- FOSS enthusiast, Debian Developer

skroutz

# About skroutz.gr

- Greece's biggest online marketplace
- Hybrid infrastructure (cloud + on-prem)
- Mostly Debian-based, mostly FOSS
- AS202042, GR-IX member since 2015

skroutz

# IXP MTU Policies

*The allowed L2 payload (L3 MTU) for GR-IX is 9000 bytes. Frames exceeding this payload may be dropped without further notice.*[1]

[1] https://www.gr-ix.gr/specs/

skroutz

*The allowed L2 payload (L3 MTU) for GR-IX is 9000 bytes. Frames exceeding this payload may be dropped without further notice.*[1]

## In practice

47 hosts with MTU 1500, 22 with MTU 9000 and 17 with something inbetween

[1] `https://www.gr-ix.gr/specs/`

skroutz

*The allowed L2 payload (L3 MTU) for GR-IX is 9000 bytes. Frames exceeding this payload may be dropped without further notice.*[1]

## In practice

47 hosts with MTU 1500, 22 with MTU 9000 and 17 with something inbetween

This must be fixed.

[1] `https://www.gr-ix.gr/specs/`

skroutz

# IXP LAN MTUs

Stats from PeeringDB:

- 1185 IXP LANs total
- 1069 (90%) with MTU 1500
- 116 (10%) with MTU 9000
  - 12 dedicated “MTU” or “Jumbo” peering LANs

skroutz

# Running mixed-MTU LANs is a bad idea

- MTU == MRU on Ethernet
- There is no signaling for frame-too-big or datagram-too-big from the receiver, just packet drops
- TCP will likely work due to TCP MSS, but *only for connections between hosts directly connected to the LAN*. Forwarded TCP traffic may suffer packet drops
- Other protocols (UDP / ICMP) will be dropped based on size

skroutz

# Running mixed-MTU IXP LANs is an even worse idea

- IXP-local traffic (i.e. BGP sessions) will work fine due to TCP MSS.
- Fragmentation and Path MTU Discovery will work partially, depending on the direction and the peers involved.
- Some (little) forwarded traffic will be dropped, increasing a couple of SNMP counters that are hard to interpret.
- The packet loss will only be noticed far away from the IXP, usually in the form of stuck TCP connections.

Finally, many stakeholders = hard coordination & troubleshooting

skroutz

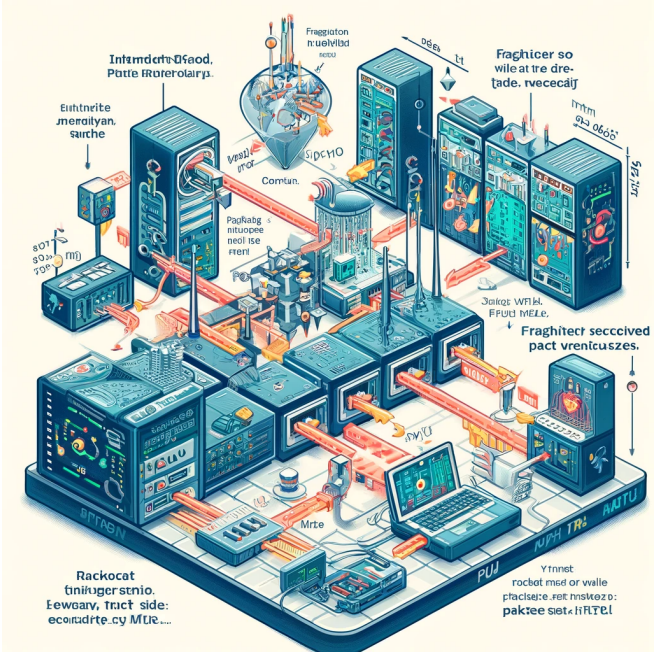
# How can IXPs transition to Jumbo Frames?

- Short answer: by making them *mandatory*
- Follow-up question: “Is it worth it?”
- No standard, only a 2011 draft (draft-mlevy-ixp-jumboframes-00[2])
- The most reliable and flexible way is running separate VLANs. This approach is favored e.g. by NETNOD
- Otherwise MTU changes *must* be coordinated between all peers at the same time

[2] https://datatracker.ietf.org/doc/html/draft-mlevy-ixp-jumboframes-00

# MTU & IP fragmentation

skroutz



Warning: RFC text follows

skroutz

# Maximum Transmission Unit

> The maximum sized datagram that can be transmitted **through the next** network is called the maximum transmission unit (MTU).
>
> If the total length is less than or equal the maximum transmission unit then submit this datagram to the next step in datagram processing; otherwise cut the datagram into two fragments *[...]*
>
> *[...]*
>
> If the Don't Fragment flag (DF) bit is set, then internet fragmentation of this datagram is NOT permitted, although it may be discarded.
>
> — RFC 791 (Internet Protocol, 1981)

skroutz

# Maximum Transmission Unit

A couple of notes on RFC 791

- MTU is defined in terms of *IPv4 datagram size*, len(IP header + payload), excluding lower-level overhead (e.g. Ethernet frame headers)
- MTU is consulted before *transmitting* the datagram. It is implicitly assumed that the other end will be able to receive it
- There is no adaptive mechanism: either fragment, or drop

skroutz

# Path MTU discovery

Fragmentation/de-fragmentation requires resources (notably buffers). To avoid fragmentation we need to detect the minimum MTU across all hops in a given network path.

skroutz

# Path MTU discovery

Fragmentation/de-fragmentation requires resources (notably buffers). To avoid fragmentation we need to detect the minimum MTU across all hops in a given network path.

> [...] we describe a technique for using the Don't Fragment (DF) bit in the IP header to dynamically discover the PMTU of a path. The basic idea is that a source host initially assumes that the PMTU of a path is the (known) MTU of its first hop, and sends all datagrams on that path with the DF bit set. If any of the datagrams are too large to be forwarded without fragmentation by some router along the path, that router will discard them and return ICMP Destination Unreachable messages with a code meaning "fragmentation needed and DF set" [7]. Upon receipt of such a message (henceforth called a "Datagram Too Big" message), the source host reduces its assumed PMTU for the path.
>
> — RFC 1191 (Path MTU Discovery, 1990)

skroutz

# Path MTU discovery

Fragmentation/de-fragmentation requires resources (notably buffers). To avoid fragmentation we need to detect the minimum MTU across all hops in a given network path.

> [...] we describe a technique for using the Don't Fragment (DF) bit in the IP header to dynamically discover the PMTU of a path. The basic idea is that a source host initially assumes that the PMTU of a path is the (known) MTU of its first hop, and sends all datagrams on that path with the DF bit set. If any of the datagrams are too large to be forwarded without fragmentation by some router along the path, that router will discard them and return ICMP Destination Unreachable messages with a code meaning "fragmentation needed and DF set" [7]. Upon receipt of such a message (henceforth called a "Datagram Too Big" message), the source host reduces its assumed PMTU for the path.
>
> — RFC 1191 (Path MTU Discovery, 1990)

Note: IPv6 prohibits fragmentation by routers and assumes a minimum MTU of 1280 bytes.

skroutz

# MTU and TCP MSS

- TCP Maximum Segment Size is used to signal the largest TCP segment a host is willing to receive
- A TCP segment is transported over an IP datagram
- Implementations set TCP MSS to (MTU - 40) for IPv4 and (MTU - 60) for IPv6
- Some middleboxes (ISP CPEs) mangle (“clamp”) MSS to the upstream MTU (usually 1492 for PPPoE)

skroutz

# We live in a 1500-byte world

```
$ tcpdump -c 5000 -i bond0 -npvl host 192.0.2.1 and tcp port 443 and 'tcp[tcpflags] = tcp-syn' \
    | grep -Eo 'mss [0-9]+' | sort -g | uniq -c
      2 mss 1286
    189 mss 1350 # IPSec tunnel mode on MTU 1500
      1 mss 1352
      1 mss 1355
      8 mss 1358
     11 mss 1360
      2 mss 1380
      4 mss 1396
      3 mss 1400
      4 mss 1412
      1 mss 1444
      1 mss 1452 # PPPoE
   4768 mss 1460 # MTU 1500
      5 mss 1560
```

skroutz

# A real case

skroutz

# Putting everything together

- Skroutz internal LAN runs on MTU 4200
- GR-IX router interfaces configured to 4200 (used to run at 9000)
- Peering with our CDN provider over GR-IX

skroutz

# An image pull gone wrong

```
14:55:55.883737 IP (tos 0x28, ttl 50, id 8675, offset 0, flags [DF], proto TCP (6), length 60)
    a.cdn.host.46162 > skroutz.server.443: Flags [S], cksum 0x4713 (correct), seq 1189807294,
    win 65520, options [mss 1560,sackOK,TS val 1766184816 ecr 0,nop,wscale 13], length 0
14:55:55.883792 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6), length 60)
    skroutz.server.443 > a.cdn.host.46162: Flags [S.], cksum 0xbe53 (incorrect -> 0x9e0b),
    seq 817513862, ack 1189807295, win 62220,
    options [mss 4160,sackOK,TS val 1507253152 ecr 1766184816,nop,wscale 8], length 0
```

skroutz

# An image pull gone wrong

```
14:55:55.883737 IP (tos 0x28, ttl 50, id 8675, offset 0, flags [DF], proto TCP (6), length 60)
    a.cdn.host.46162 > skroutz.server.443: Flags [S], cksum 0x4713 (correct), seq 1189807294,
    win 65520, options [mss 1560,sackOK,TS val 1766184816 ecr 0,nop,wscale 13], length 0
14:55:55.883792 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6), length 60)
    skroutz.server.443 > a.cdn.host.46162: Flags [S.], cksum 0xbe53 (incorrect -> 0x9e0b),
    seq 817513862, ack 1189807295, win 62220,
    options [mss 4160,sackOK,TS val 1507253152 ecr 1766184816,nop,wscale 8], length 0
```

- Two hosts, far away from GR-IX, agreed to try to pass a 1600-byte IP datagram to each other
- Our network had a 1600-byte clean path to GR-IX; fragmentation did not kick in
- The CDN's GR-IX router had a 1500-byte MTU, dropping the packet and causing the connection to hang (affecting end-users).
- Now we are forced to switch over to 1500-byte MTU to get things working

skroutz

# Hack-of-the-day: per-destination MTU?

skroutz

# Linux route MTU attribute

- Linux has per-route MTU attributes
- We can abuse this to enforce different MTUs when forwarding to different IX peers
- Or follow the Postel principle: receive packets up to 9k bytes, but only send out 1500-byte ones

skroutz

# Sample BIRD config

```
# Assuming interface MTU has been configured to 9000

protocol kernel grix_v4 {
  ipv4 {
    export filter {
      # Set MTU for all routes going through a specific peer
      if gw = 192.0.2.42 then krt_mtu = 1500;
      ...
      # or for all routes going over GR-IX
      if iface = "grix" then krt_mtu = 1500;
    };
  };
}
```

skroutz

# Thank you!

https://engineering.skroutz.gr/

skroutz