Fragmentation at Network Layer

- Difficulty Level : Medium
- Last Updated : 06 Jul, 2021

Prerequisite – IPv4 Datagram Fragmentation and Delays

Fragmentation is done by the network layer when the maximum size of datagram is greater than maximum size of data that can be held a frame i.e., its Maximum Transmission Unit (MTU). The network layer divides the datagram received from transport layer into fragments so that data flow is not disrupted.

- Since there are 16 bits for total length in IP header so, maximum size of IP datagram = $2^{16} - 1$ = 65, 535 bytes.

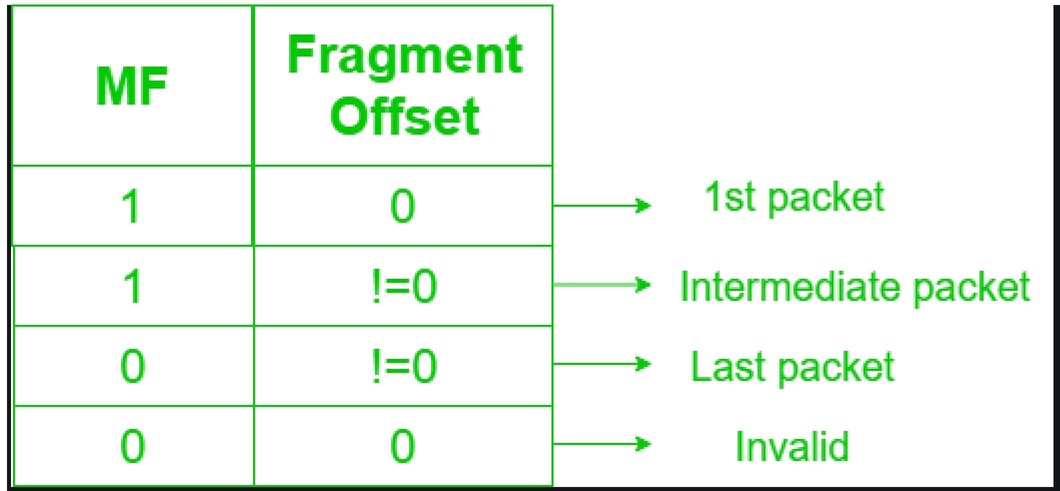
- It is done by network layer at the destination side and is usually done at routers.
- Source side does not require fragmentation due to wise (good) segmentation by transport layer i.e. instead of doing segmentation at transport layer and fragmentation at network layer, the transport layer looks at datagram data limit and frame data limit and does segmentation in such a way that resulting data can easily fit in a frame without the need of fragmentation.
- Receiver identifies the frame with the **identification (16 bits)** field in IP header. Each fragment of a frame has same identification number.
- Receiver identifies sequence of frames using the **fragment offset(13 bits)** field in IP header
- An overhead at network layer is present due to extra header introduced due to fragmentation.

Fields in IP header for fragmentation –

- **Identification (16 bits) –** use to identify fragments of same frame.
- **Fragment offset (13 bits) –** use to identify sequence of fragments in the frame. It generally indicates number of data bytes preceding or ahead of the fragment.
  Maximum fragment offset possible = (65535 – 20) – 1 = 65514
  {where 65535 is maximum size of datagram and 20 is minimum size of IP header}
  So, we need ceil($\log_2 65514$) = 16 bits for fragment offset but fragment offset field has only 13 bits. So, to represent efficiently we need to scale down fragment offset field by $2^{16}/2^{13} = 8$ which acts as a scaling factor. Hence, all fragments except the last fragment should have data in multiples of 8 so that fragment offset $\in$ N.
- **More fragments (MF = 1 bit) –** tells if more fragments ahead of this fragment i.e. if MF = 1, more fragments are ahead of this fragment and if MF = 0, it is the last fragment.
- **Don't fragment (DF = 1 bit) –** if we don't want the packet to be fragmented then DF is set i.e. DF = 1.

Reassembly of Fragments –

It takes place only at destination and not at routers since packets take independent path(datagram packet switching), so all may not meet at a router and hence a need of fragmentation may arise again. The fragments may arrive out of order also.



MF	Fragment Offset	
1	0	1st packet
1	!=0	Intermediate packet
0	!=0	Last packet
0	0	Invalid

Algorithm –

1. Destination should identify that datagram is fragmented from MF, Fragment offset field.
2. Destination should identify all fragments belonging to same datagram from Identification field.
3. Identify the 1st fragment(offset = 0).
4. Identify subsequent fragment using header length, fragment offset.
5. Repeat until MF = 0.

Efficiency –
Efficiency (e) = useful/total = (Data without header)/(Data with header)
Throughput = e * B { where B is bottleneck bandwidth }

Example – An IP router with a Maximum Transmission Unit (MTU) of 200 bytes has received an IP packet of size 520 bytes with an IP header of length 20 bytes. The values of the relevant fields in the IP header.
Explanation – Since MTU is 200 bytes and 20 bytes is header size so, the maximum length of data = 180 bytes but it can't be represented in fragment offset since it is not divisible by 8 so, the maximum length of data feasible = 176 bytes.
Number of fragments = (520/200) = 3.
Header length = 5 (since scaling factor is 4 therefore, 20/4 = 5)
Efficiency, e = (Data without header)/(Data with header) = 500/560 = 89.2 %

	20 | 176	20 | 176	20 | 148	[redacted]
Fragment Offset	0	22	44	[redacted]
MF	1	1	0	[redacted]
Header length	5	5	5	[redacted]
Total length	196	196	168	[redacted]