Fragmentation

By Michael Almandeel

Fragmentation is the process by which a datagram payload is broken into smaller pieces, and sent as a series of smaller datagrams. This is necessary because in most cases, the size of the payload plus the ip header and frame header exceeds the mtu(maximum size of a frame) of the data link on which the frames will be forwarded.

Fragmentation will occur , as needed(explained above) in either the host who sends the datagram, or any router which forwards the datagram on its path to the target host. The process of fragmentation occurs as follows:

-the new payload size is calculated as mtu – frame header length – datagram header length.

-the payload size of the datagram is divided by new payload size, and 1 is added to this number if there is a remainder. This number represents how many new datagrams will be sent. Its important to note that these new datagrams will each carry a piece of the original datagram payload.

-a sequence of frames(all with the same frame header) is created(the number of frames is identical to the number explained directly above), and the payload of each frame is a new datagram. Each of the datagrams is comprised of a payload of length new payload size, in which the payload is equal to a subsequence of the original payload, wherein the new sequence begins at originalPayload[sequenceNum*newPayloadSize], and a new datagram header, which is identical to the original header, with the exceptions that the fragment field is equal to the fragment field of the original header plus the product of (sequence number * new payload size), and the more flag is copied to all datagram headers if 1, or if 0, 1 is copied to all the more flags except the final datagram in the sequence, in which 0 will be copied to the more flag.

-the sequence of frames created in the previous step is forwarded out the interface which corresponds to the hardware address in the frame headers.

Note that this process is repeated anytime the mtu of the outgoing interface is smaller than the mtu of the incoming interface.

When the target host finally receives one of these datagrams, it allocates a 64k buffer in memory for the number specified in the identification field, and sets a timer. If at any point it receives a datagram where the more flag is 0, it deallocates all bytes in the buffer beyond the fragment offset of that datagram plus the length of the datagram, and copies the payload of that datagram starting at the fragment offset. If the more flag is 1, it simply places the payload of the datagram in the buffer, starting at the location in the buffer specified by the fragment offset. If all datagrams which correspond to this identification field are not received by the time the timer expires, the entire buffer is deallocated and the data received thus far is lost. If all of the datagrams are received, this data can now be sent to a higher layer protocol for processing.

For example, Imagine host A wishes to send a 64k datagram to host B. In between host A and B, there exists router A. The interface on router A which is connected to host A has an mtu of 1500 bytes, and the interface of router A which is connected to host B has an mtu of 750 bytes. Host A checks the mtu for its data link to router A, and realizes that the 64k datagram(plus the frame header) it wishes to send to host B is larger than 1500 bytes. Host A executes the logic described above and breaks the payload of the original datagram into pieces, wherein each piece of the payload is concatenated with a new frame header and datagram header. All of these frames are then sent to router A on the corresponding data link. Router A, upon receiving these frames, realizes that the mtu of the data link that corresponds to host B is smaller than the frames is must forward, and router A repeats the same process that host A did, breaking the payload into sub payloads and concatenating a new frame and datagram header to each new payload. These are then sent to host B on its corresponding data link. Host B receives one of these frames, and allocates a 64k buffer, also setting a timer. It begins copying payloads to its buffer as described above, and if the buffer is saturated before the timer expires, this data can be passed to a higher level protocol for processing. Otherwise, host B will destroy the buffer and have no recollection that any of that data existed at all. If host A wishes for B to receive the data, it must send everything all over again.